ANVM - Virtual Mapping

Proyecto de Sistemas Informáticos
Facultad de Informática, UCM
Curso 2012/2013

Autores:
Alejandro Peñalver Santorio
Ricardo Pragnell Valentín

Director:
Juan Carlos Fabero Jiménez

Co-directora:
Guadalupe Miñana Ropero

Autorización

Se autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado.

Firmado:

Alejandro Peñalver Santorio

Ricardo Pragnell Valentín

Resumen

El mapeado y la localización en interiores han sido un problema, en su mayor parte, no resuelto. Las técnicas actuales utilizan escáneres láser 3D (LIDAR) y posicionamiento mediante ondas de radio. Sin embargo, uno de los principales inconvenientes que tienen estas técnicas es el empleo de equipos muy costosos que requieren mucho tiempo para su despliegue y utilización. En este proyecto se presenta una solución tanto hardware como software: ANVM. Proporciona un dispositivo de mapeado económico y preciso, software para la edición de mapas y una aplicación para smartphones basada en Android. Nuestro dispositivo para realizar el escáner está compuesto por un Microsoft Kinect, un microprocesador Arduino Uno y una plataforma móvil personalizada que utiliza codificadores rotatorios digitales para llevar un seguimiento de la posición. Combinando los datos provenientes de la nube de puntos del Kinect y la información de la posición podemos virtualizar espacios interiores utilizando el algoritmo Rao-Blackwellized Particle Filter SLAM. Tras muchas y exhaustivas pruebas, hemos conseguido buenos resultados y alcanzado nuestro objetivo de crear un dispositivo de mapeado a un coste muy reducido.

Palabras clave: mapas 2D, Android, localización interiores, código QR, panorama, SLAM, Kinect, reconstrucción, virtualización, rutas.

Abstract

Indoor localization and mapping has been a largely unresolved problem. Current state-of-the-art approaches use 3D laser scanning (LIDAR) and radio wave positioning. A major drawback is they require very expensive equipment and are time consuming to set up. We present a complete software and hardware solution: ANVM (Augmented Navigation & Virtual Mapping). Providing an inexpensive but accurate mapping device, map editing software and a smartphone application based on Android. Our scanner device is comprised by a Microsoft Kinect, an onboard Arduino Uno microprocessor and a custom mobile platform which utilizes digital encoders for position tracking. Combining the Kinect point cloud data and position information we are able to virtualize indoor environments utilising the Rao-Blackwellized Particle Filter SLAM technique. Through extensive experiments, we have achieved good results succeeding in our goal to create an inexpensive mapping device.

Keywords: 2D Maps, Android, indoor localization, QR code, panorama, SLAM, Kinect, reconstruction, virtualization, routes.

Índice

1 Introducción 1

1. ¿Qué es ANVM? 1

2. Idea inicial 2

3. Sub-proyectos de ANVM 2
3.1. ANVM: Mobile app . . . 3
3.2. ANVM - Map Editor . . . 4
3.3. ANVM: Virtual Mapping . . . 5

2 Estado del arte y visión 7

1. Posicionamiento 7
1.1. Planteamiento del problema . . . 7
1.2. Declaración de posición del producto . . . 9

2. Interesados y descripción de usuarios 10
2.1. Demografía de mercado . . . 10

3. Alternativas y competidores 12
3.1. Viametris . . . 12
3.2. Matterport . . . 13
3.3. Google . . . 14

4. Visión general del producto 19
4.1. Perspectiva del producto . . . 19
4.2. Sumario de capacidades . . . 20

5. Características del producto 21

3 Investigación 23

1. SLAM: Simultaneous localization and mapping 24
1.1. 3D Slam . . . 24
1.2. 2D Slam . . . 26

2. Localización basada en imágenes 27
2.1. Keypoints . . . 27
2.2. Structure-from-Motion . . . 31
2.3. Estimación de la posición . . . 34

3. Librerías y Proyectos investigados 35
3.1. PCL : Point Cloud Library . . . 35
3.2. ACG_LOCALIZER . . . 39

3.3. ROS: Robot Operating System 43
3.4. Arduino 46
3.5. MRPT: Mobile Robot Programming Toolkit 48

4. Conclusiones **53**

4 Requisitos 55

1. Descripción general **55**
1.1. Perspectiva del producto 55
1.2. Funcionalidades del producto 55
1.3. Características de los usuarios 56
1.4. Limitaciones 56
1.5. Supuestos y dependencias 57

2. Requisitos específicos **57**
2.1. Funcionalidad 57
2.2. Usabilidad 58
2.3. Seguridad 59
2.4. Rendimiento 59
2.5. Restricciones del diseño 59
2.6. Componentes adquiridos 60

5 Plan de desarrollo 61

1. Descripción del proyecto **61**
1.1. Propósito del proyecto, alcance y objetivos 61
1.2. Suposiciones y limitaciones 62
1.3. Entregas del proyecto 63
1.4. Evolución del plan de desarrollo de software 64

2. Organización del proyecto **66**
2.1. Estructura organizativa 66
2.2. Colaboradores externos 66
2.3. Roles y responsabilidades 67

3. Proceso de gestión **69**
3.1. Estimaciones 69
3.2. Plan de proyecto 69
3.3. Planes de iteración 74
3.4. Control y seguimiento del proyecto 83

6 Proceso de reconstrucción 87

1. Conceptos teóricos **87**
1.1. Captura de datos de odometría 87
1.2. Captura de datos de la imagen 90

1.3. Algoritmo de reconstrucción 91

2. Cómo llevar a cabo una reconstrucción 93
2.1. Captura de datos 93
2.2. Generación del mapa 95

7 Arquitectura 97

1. Objetivos y restricciones de la arquitectura 97
1.1. Objetivos 97
1.2. Restricciones 98

2. Arquitectura del robot 99
2.1. Construcción Plataforma Móvil: 99

3. Software de reconstrucción y captura de datos 102
3.1. Vista lógica 104
3.2. Vista de procesos 113

4. Vista de despliegue 115

8 Plan de pruebas 117

1. Requisitos de las pruebas 117

2. Objetivos de las pruebas 118

3. Pruebas 118
3.1. Primera versión: Kinect sin uso de odometría 119
3.2. Segunda versión: inclusión de odometría 120
3.3. Tercera versión: mejoras físicas en el robot 120

4. Conclusiones 123

9 Glosario 125

10 Bibliografía y referencias 133

Capítulo 1
Introducción

Este capítulo del documento trata de introducir al usuario en el contexto del proyecto ANVM, intentando aclarar los aspectos principales del mismo. Durante todo el documento se utilizarán palabras con las que el lector puede no estar familiarizado. Para cualquier consulta sobre vocabulario se puede consultar la sección Glosario.

1. ¿Qué es ANVM?

ANVM es un proyecto novedoso que trata de incurrir en los campos de la robótica, computación visual, y desarrollo de aplicaciones móviles. El proyecto en su conjunto trata de aportar una solución al problema de la localización en interiores. Cuántas veces nos hemos sentido perdidos en un centro comercial, no sabiendo hacia que lado ir para buscar unos aseos, o una cierta tienda, o en un aeropuerto, cuántas veces hemos deseado encontrar sin problemas una puerta de embarque o un restaurante donde comer algo. ANVM trata de ofrecer una solución a estos problemas permitiendo localizarse dentro de un espacio interior.

Por tanto, este proyecto pretende ofrecer a un espacio interior concreto (centros comerciales, aeropuertos, universidades) una forma de que sus usuarios se orienten mejor en dichos espacios, siguiendo para esto un proceso muy simple.

1. En primer lugar de la reconstrucción virtual del espacio para la obtención de un mapa en 2D (si se dispone ya de uno, esta fase puede no ser necesaria).
2. En segundo lugar un proceso de recogida de información sobre los puntos que se desean mostrar sobre este mapa.

Para ello, este proyecto se divide a su vez en proyectos más pequeños que realizan cada una de estas tareas. ANVM-Virtual Mapping y ANVM-Mobile App.

ANVM - Virtual Mapping se encarga del preprocesamiento del espacio interior. Permite la obtención de un mapa de dicho espacio, que después se podrá utilizar en la aplicación móvil para la ofrecer al usuario final la funcionalidad requerida.

ANVM - Mobile App permite localizarse dentro de un espacio interior concreto, el cuál ha tenido que pasar antes un proceso previo de preparación comentado anteriormente. La localización se lleva a cabo mediante un smartphone, dispositivo del que actualmente disponen la mayoría de los usuarios. A través del mismo este proyecto ofrece un conjunto de funcionalidades de gran utilidad para el usuario dentro de ese espacio interior.

En este proyecto también se incluye un programa desarrollado en Java ANVM - Map Editor que actúa como nexo entre los dos “sub-proyectos”. Esta aplicación para ordenador permite que los datos generados por la reconstrucción virtual puedan ser interpretados después por la aplicación móvil. Además permite la inserción de datos sobre los puntos de interés que desee el cliente del proyecto que aparezcan dentro de la aplicación móvil.

Unidos estos proyectos (Virtual Mapping + Mobile app + Map Editor) ofrecen el conjunto de funcionalidad de ANVM formando un proyecto completo. Cabe destacar que a pesar de ser dos proyectos separados, están totalmente relacionados en la medida en la que uno depende del otro y viceversa. Como se detallará en las siguientes secciones, los cambios en un proyecto afectan al otro, y por tanto a lo largo de este documento se mencionará el proyecto paralelo (ANVM - Mobile app) a éste que aquí se trata.

2. Idea inicial

ANVM surgió con la idea principal de sacar partido a un dispositivo que ofrece grandes posibilidades, y este dispositivo es el Kinect de Microsoft. Según conocimientos previos a este proyecto, se sabía que este dispositivo era capaz de virtualizar un espacio, y reconocer objetos dentro de él. A partir de este concepto, se pensó aplicarlo a un problema de la vida real: la localización en interiores.

La primera intención era conseguir virtualizar tridimensionalmente el espacio interior con el Kinect y, posteriormente, poder utilizar dicha reconstrucción virtual para localizar usuarios dentro de él. Con el creciente uso de los smartphones en nuestra sociedad, parecía una buena idea realizar esa localización mediante estos dispositivos de uso tan común. Una vez el usuario estuviera localizado dentro de cualquier punto del espacio interior, la intención era poder guiarle a través de la reconstrucción hasta llegar al sitio que el usuario deseara.

Esta primera idea se ha mantenido en esencia, el proyecto ANVM sigue ofreciendo esa funcionalidad, pero de una manera distinta. Se ofrece una virtualización en dos dimensiones (un plano o mapa) en vez de una reconstrucción tridimensional. Se ofrece también la posibilidad de navegar en el smartphone por la reconstrucción, pero no de forma tridimensional sino bien mediante el uso del mapa, o de imágenes panorámicas.

La localización del proyecto actual difiere también de la idea inicial, que era poder localizarse en cualquier punto del espacio interior. Esto se pretendía que fuera realizando una foto a cualquier punto del espacio interior, y mediante la técnica de localización basada en imágenes ser capaz de establecer la posición del usuario dentro de la reconstrucción. Esto no fue posible por motivos que se detallarán en la sección de Investigación. En su lugar se utilizó la técnica QR, esto es, emplazar dentro del espacio interior sobre el que nos queremos localizar un comjunto de “pegatinas” con códigos QR que sirvan para identificar cada punto de la reconstrucción.

3. Sub-proyectos de ANVM

A continuación se presenta una introducción a cada uno de los proyectos que forman ANVM con la intención de dejar clara qué parte de funcionalidad aporta cada uno, y que problema concreto resuelven. En la figura 11 podemos ver una vista general del sistema completo.

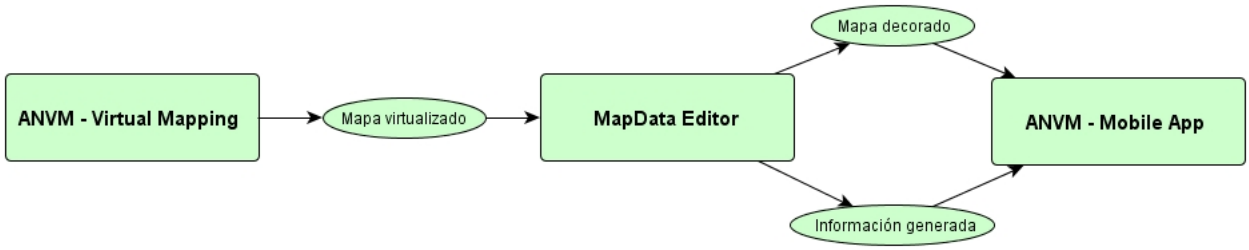

Figura 1: Vista general ANVM

3.1. ANVM: Mobile app

Se trata del proyecto paralelo a éste. Es el encargado del desarrollo de la aplicación móvil. Esta aplicación implementada para smartphones Android (aunque en un futuro puede ser ampliable a iOS) se encarga de ofrecer toda la funcionalidad para el usuario final.

Figura 2: ANVM: Mobile App

A partir del mapa generado por ANVM - Virtual Mapping, y después de recibir la información necesaria proveniente del ANVM - Map Editor, permite al usuario navegar por dicho mapa. Ofrece además un conjunto de utilidades que se detallan a continuación:

- **Localización**: Permite establecer la posición del usuario a partir de la utilización de códigos QR repartidos por todo el espacio interior.
- **Guiado**: Una vez localizado el usuario, se le permite elegir de entre una lista de puntos de interés a los que poder dirigirse, que dependerán del espacio interior concreto. Además se le ofrece una guía mediante el mapa 2D o a través de imágenes panorámicas que corresponden con una vista más natural del espacio reconstruido.

- **Exploración del mapa**: Se ofrece la opción de navegar por el mapa del sitio, o por sus imágenes panorámicas (estilo Google Street View) para que el usuario pueda descubrir los puntos de interés del lugar virtualizado.

Esta es una aplicación totalmente general que funciona para cualquier espacio virtualizado. Es decir, los parámetros de los que depende son el mapa del sitio y sus puntos de interés (restaurantes, aseos, despachos o salidas, por mencionar algunos) que dependen del lugar que se ha reconstruido. Está diseñada para que la aplicación se configure automáticamente dependiendo del lugar, sin necesidad de crear una aplicación móvil para cada lugar que se reconstruya. Si, por ejemplo, se realiza la virtualización de un aeropuerto, se generarán los archivos necesarios mediante la herramienta Map Editor para que la aplicación se configure correctamente y muestre tanto el mapa como los datos del aeropuerto. Si se decide reconstruir un centro comercial se sigue el mismo proceso sin necesidad de modificar la aplicación móvil más allá de un cambio en la estética y rótulos.

3.2. ANVM - Map Editor

Es un programa desarrollado por los integrantes del proyecto ANVM que sirve de enlace entre este proyecto y la aplicación móvil. Su intención es permitir que el mapa virtualizado conseguido a partir de la aplicación de este proyecto sirva para poder incrustarlo en la aplicación móvil junto con un conjunto de datos sobre el espacio reconstruido. Posteriormente estos datos permitirán a la aplicación móvil ofrecer la funcionalidad descrita anteriormente.

Cabe destacar que el usuario objetivo de este programa no es el usuario final del proyecto, sino el usuario intermedio para el que se está particularizando la aplicación. Esta herramienta permite la configuración de los parámetros del mapa obtenido mediante la reconstrucción con ANVM-Virtual Mapping y esos datos han de ser proporcionados por dicho usuario, por ejemplo el dueño del centro comercial o del aeropuerto que se esté reconstruyendo. En su defecto, el usuario serían los desarrolladores de este proyecto o el que haya realizado la reconstrucción del espacio interior, finalizando el proceso de configuración aportando los parámetros necesarios a esta aplicación Java.

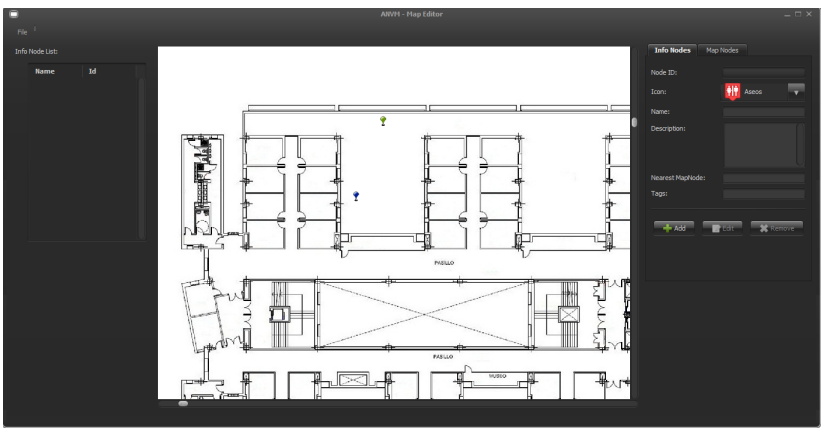

Figura 3: ANVM: Map Editor

Está desarrollado en Java, lenguaje altamente conocido por los integrantes del proyecto gracias a las numerosas asignaturas de la carrera en las que se utiliza. Se hablará más adelante en el resto de secciones más detalladamente de esta aplicación, pero se van a enumerar ahora el conjunto de funcionalidades que pretende ofrecer:

- **Inserción de puntos de interés**: dado que una de las funcionalidades de la aplicación móvil es ofrecer al usuario el conjunto de puntos de interés de los que dispone el lugar, se han de indicar en algún punto a la aplicación cuáles son esos puntos. Este programa permite la localización en el mapa de dichos puntos así como dotar a cada uno de ellos de un título y una descripción del punto de interés y elegir el icono con el que posteriormente aparecerá este punto en la aplicación móvil.
- **Inserción de puntos clave del mapa**: dado que se pretende guiar al usuario mediante la aplicación móvil, este programa permite la inserción de puntos clave (nodos) en el mapa que conformarán el grafo de posibles puntos a los que se puede llegar y desde los que se puede localizar el usuario. Habrá que insertar tantos puntos como códigos QR haya localizados en el espacio interior, y más si fueran necesarios. Esta tarea corresponde a la configuración de la aplicación y por tanto la debería realizar el técnico que esté instalando ANVM en el espacio interior.
- **Conexiones**: permite seleccionar los nodos del mapa que están unidos unos con otros, es decir aquellos a los que se puede llegar desde un cierto nodo. Esta información es necesaria para el posterior trazado de rutas en la aplicación móvil por las que guiar al usuario final a través del mapa.
- **Distancias**: dado que el mapa está a escala o bien reconstruido por la técnica SLAM que ofrece este proyecto, las distancias que calcula esta aplicación son reales y corresponden con las distancias entre cada uno de los nodos del mapa.

Tras recolectar toda la información necesario mediante esta aplicación se generará un fichero XML que será posteriormente enviado a la aplicación móvil para que contenga los datos necesarios y el proyecto ANVM esté completamente configurado. En la figura 3 podemos encontrar una captura sobre esta aplicación.

3.3. ANVM: Virtual Mapping

Esta parte del proyecto ANVM es a la que está dedicada el resto del documento. Habrá secciones en las que se haga referencia y se incluya información adicional sobre ANVM - Map Editor ya que cierta parte forma parte del desarrollo de este proyecto y está directamente relacionada con la virtualización, así como a ANVM - Mobile app dado que es el producto final para el que se realiza todo el proceso de mapeado.

El objetivo principal de este subproyecto dentro de ANVM es realizar una reconstrucción (SLAM) de un cierto espacio interior para generar un mapa 2D de dicho lugar que pueda servir de guía para la aplicación móvil. Para la realización de esta reconstrucción se utilizan ciertos componentes hardware, que unidos forman un dispositivo capaz de realizar este cometido (ver figura 4). El técnico o desarrollador de ANVM que esté realizando la reconstrucción tendrá que guiar el dispositivo por toda la instalación que

se desee virtualizar de forma manual, de forma que el dispositivo vaya reconstruyendo el espacio a medida que se va avanzando por el mismo.

Figura 4: Dispositivo ANVM capaz de realizar la virtualización

Dichos componentes hardware son principalmente:

- **Sensor Kinect de Microsoft**, capaz de captar el espacio y generar nubes de puntos que lo describen, y un "robot" capaz de ofrecer información sobre la posición actual del Kinect dentro de la reconstrucción.
- Un robot formado por varios componentes:
  - **Placa Arduino:** necesaria para la comunicación con el ordenador cuando se están recogiendo los datos en una virtualización.
  - **Dos encoders:** que proporcionan información sobre la distancia que se ha movido el robot así como la dirección en la que se está moviendo.
  - **Carro portátil:** empleado para sostener el arduino junto con el Kinect y los encoders instalados en las ruedas.

ANVM - Virtual mapping ofrece por tanto una manera sencilla de obtener un mapa 2D a escala de un espacio interior, algo muy útil cuando no se dispone de uno, y mediante un simple "paseo" por las instalaciones. El dispositivo utilizado para realizar este SLAM ha sido construido por los desarrolladores de este proyecto, y ofrece una alternativa bastante económica para realizar virtualizaciones de un cierto lugar interior. Se contempla también en un futuro poder realizar virtualizaciones en tres dimensiones, aunque de momento esa funcionalidad no está implementada en este proyecto.

Durante el resto del documento se realizará una descripción más exhaustiva de todo el proceso de desarrollo y construcción necesario para que ANVM - Virtual Mapping haya sido posbile, así como su largo proceso de investigación que marcó el devenir del proyecto.

Capítulo 2

Estado del arte y visión

El propósito de este capítulo del documento es recoger, analizar y definir las necesidades y espectativas a alto nivel del proyecto ANVM. Al tratarse de un proyecto conjunto con la aplicación móvil, se plantearán las necesidades que pretende cubrir el proyecto ANVM en su conjunto, tratando de profundizar un poco más en ciertas secciones en este subproyecto que es Virtual Mapping. Se centra en las capacidades requeridas por los posibles interesados en el proyecto así como en los usuarios a los que está destinado el producto. Todos los términos científicos, tecnicismos, siglas y demás expresiones que necesiten una breve explicación complementaria están recogidos en la sección Glosario.

1. Posicionamiento

En esta sección se describen mediante tablas los principales problemas identificados y en consecuencia se hará una declaración de posición del proyecto.

1.1. Planteamiento del problema

En las siguientes tablas se ilustran los problemas o necesidades detectadas a los que ANVM tratará de dar solución con sus correspondientes impactos y soluciones.

El problema de	La localización en interiores.
Afecta a	Usuarios que necesitan conocer su posición dentro de un lugar cubierto donde no existe señal GPS para localizarse.
Impacto	El usuario tendrá que dirigirse a un punto de información para poder consultar un mapa. Cabe la posibilidad de que este punto de información ni siquiera exista en algunos lugares. Es posible también que a pesar de existir dicho mapa, no esté localizado en él el usuario.
Solución	ANVM ofrece la posibilidad de que el usuario se localice en el mapa virtual capturando un código QR.

Tabla 1: Descripción del problema de la localización en interiores

El problema de	La obtención de indicaciones para dirigirse a un lugar en interiores.
Afecta a	Usuarios que se encuentran en espacios cubiertos desconocidos donde no hay señal GPS y necesitan llegar a algún lugar de dicho espacio.
Impacto	Si el usuario cuenta con un mapa, debe ser capaz de orientarse en él y consultarlo periódicamente a medida que avanza en su recorrido. En caso de que no cuente con un mapa, el usuario tendrá que dirigirse a un punto de información para poder consultar un mapa y memorizar el recorrido que debe hacer. Cabe la posibilidad de que este punto de información ni siquiera exista en algunos lugares.
Solución	ANVM ofrece al usuario la ruta óptima para llegar al lugar que desee. Para ello recibe indicaciones incorporadas en el mapa que se está mostrando en la pantalla de su dispositivo móvil.

Tabla 2: Descripción del problema de obtención de indicaciones

El problema de	La no existencia de un mapa del lugar en el que se encuentra el usuario.
Afecta a	Usuarios que se encuentran en espacios cubiertos desconocidos y de los cuales no tiene ningún tipo de mapa con el que orientarse.
Impacto	El usuario tendrá que dirigirse a un punto de información para poder consultar un mapa. Cabe la posibilidad de que este punto de información ni siquiera exista en algunos lugares.
Solución	ANVM ofrece la posibilidad de que el usuario consulte el mapa en el dispositivo móvil, pudiendo obtener más detalle empleando zoom. Además incluye información de los puntos de interés más importantes marcados sobre el propio mapa, con descripciones sobre los mismos.

Tabla 3: Descripción del problema de la consulta de un mapa virtual

El problema de	La búsqueda de puntos de interés
Afecta a	Usuarios que se encuentran en espacios cubiertos desconocidos, que están buscando algún punto de interés como despachos, baños públicos o cafeterías
Impacto	El usuario tendrá que dirigirse a algún empleado del lugar o algún mapa para recibir indiaciones.
Solución	ANVM ofrece la posibilidad de que el usuario consulte el mapa en el dispositivo móvil, pudiendo realizar una búsqueda dentro de la propia aplicación. Una vez seleccionado el punto de interés la aplicación muestra información sobre ella y da la posibilidad de activar las indicaciones para dirigirse hasta allí.

Tabla 4: Descripción del problema de la consulta de un mapa virtual

1.2. Declaración de posición del producto

En el cuadro 5 podemos ver la declaración de la posición del proyecto ANVM - Virtual Mapping.

Para	Todos aquellos usuarios o propietarios de un cierto espacio interior
Que	Deseen obtener un mapa de dicho espacio, ya sea de su propia casa, de su centro comercial, ministerio...
ANVM	Es una suite de herramientas que proporciona una aplicación en conjunto con un dispositivo hardware para llevar a cabo una reconstrucción del lugar, para obtener dicho mapa, y posteriormente permitir la utilización del mapa en la aplicación móvil, con todas las ventajas que ésta ofrece como, por ejemplo, realizar búsquedas de puntos de interés que se encuentren en él.
Que	Gracias a la cual el usuario nunca se encontrará desorientado.
Al contrario de	Otras aplicaciones que utilizan la señal GPS, la cual no está disponible en interiores, y que no proporcionan orientación al usuario.
Nuestro proyecto	Se presenta como una alternativa económica para empresas que busquen promocionar su negocio mediante el uso de las nuevas tecnologías o necesiten dar un soporte de guía por sus instalaciones a los usuarios.

Tabla 5: Declaración de posición de producto

2. Interesados y descripción de usuarios

En esta sección se va a describir detalladamente la demografía de mercado en el que se movería ANVM, así como una descripción de los perfiles de los usuarios finales y los interesados.

2.1. Demografía de mercado

Este proyecto trata de cubrir una necesidad en el mercado de la virtualización de espacios, aunque no hay que olvidar que ANVM - Virtual Mapping es un subproyecto de ANVM que trata de cumplir un cometido dentro la funcionalidad total del proyecto y que, por tanto, el usuario final del proyecto ANVM es el propietario de un smartphone que desea orientarse en un espacio interior. En este apartado nos vamos a centrar en el mercado del SLAM, que es al que está orientado este subproyecto. Para consultar datos sobre el mercado de usuarios finales de ANVM se puede consultar el documento del proyecto paralelo ANVM - Mobile app.

En primer lugar es necesario comprender que este mercado se ha comenzado a explotar desde hace poco tiempo, debido a los avances que se han ido sucediendo en el campo de la visión por computador, robótica, etc. Por tanto es un mercado actualmente en expansión, en el que no se conoce exactamente el número de usuarios o interesados que puede tener. Las técnicas de virtualización y localización de espacios (SLAM) se aplican en multitud de campos, y continuamente surgen nuevos campos que necesitan de la funcionalidad que el SLAM ofrece. Gracias a los avances en los últimos años se ha conseguido reducir considerablemente el tamaño de los procesadores a la vez que aumentar su capacidad de procesamiento. Esto abre las puertas a un mercado más orientado hacia el usuario común. Aunque hasta hace pocos años esta tecnología estaba limitada a usos en su mayoría científicos o industriales, la adopción al mercado comercial se esta llevando a cabo a un ritmo acelerado, ya que los usuarios demandan, cada vez más, nuevas formas de interactuar con sus dispositivos y entorno. SLAM y otras técnicas similares rompen con la frontera de interacción 2D, ya obsoleta, dando paso a una nueva era de interacción 3D. A continuación se exponen algunos ejemplos de los campos que intenta abarcar este mercado.

Automovilística

En este campo se está investigando actualmente con las técnicas de SLAM, y ya se están fabricando los primeros coches que utilizan esta tecnología. Sus aplicaciones en este mercado se mueven desde el reconocimiento de obstáculos para poder evitarlos, o asistir en la frenada al conductor, hasta el reconocimiento de la carretera, que permita al coche conducir de forma automática sin necesidad de conductor.

Robótica

Este es el mercado en el que se aplican la mayoría de estas técnicas y se está produciendo un gran aumento en las inversiones para investigación en este campo. Es obvia la necesidad por parte de los robots de reconocer el espacio por el que se están moviendo para ser capaces de hacerlo de forma automática. No existen datos exactos sobre el número de robots que emplean estas técnicas dado que se están aplicando en estos últimos años, y la mayoría todavía no han salido al mercado.

Seguridad e investigaciones

En el campo de investigación forense sería de gran utilidad tener una virtualización de la escena del crimen. Una reconstrucción detallada puede ser estudiada con más detenimiento por investigadores sin necesidad de encontrarse físicamente en la zona. Además puede emplearse como una prueba más fehaciente que una simple fotografía en un juicio.

Entretenimiento y videojuegos

El sector de videojuegos en los últimos años se ha convertido en una de las actividades económicas en mayor desarrollo, superando con creces los beneficios de la industria del cine. Los consumidores buscan nuevas experiencias y una forma de ofrecerlas es empleando tecnologías como SLAM. Mediante técnicas de virtualización se pueden generar niveles del videojuego emplazados en los domicilios, vecindarios y otros lugares conocidos por el usuario, permitiendole jugar dentro de estas mismas reconstrucciones.

Conservación de patrimonio histórico

Otra oportunidad de mercado es la vitualización de patrimonio histórico como anfiteatros, plazas antiguas y sendas protegidas, entre otras. Disponer de una reconstrucción de antiguas construcciones supone la conservación, al menos en forma digital, para próximas generaciones.

Localización en interiores

Esta parte del mercado es a la que esta parte del proyecto de ANVM trata de aportar funcionalidad. Actualmente existen muy pocas empresas que se dedican a esto, a la generación de mapas y localización en interiores. El mercado que aquí se trata de explotar es el de aquellos propietarios que deseen obtener de una forma sencilla un mapa de su espacio interior. Podemos pensar en lugares grandes como aeropuertos, centros comerciales, o en lugares más pequeños como oficinas o pequeños locales. A continuación se van a presentar algunos datos sobre el número de espacios interiores a los que ANVM puede ofrecer su funcionalidad:

- Según datos del 2012 existen en España más de 15.3 millones de m^2de superficie de centros comerciales, que pueden ser reconstruidos con la aplicación ANVM - Virtual mapping, repartidos en un total de 670 centros por todo el país. Actualmente se están implantando gran variedad de nuevas tecnologías en estos centros que permitan al usuario una visita más cómoda al mismo y ANVM cumple con estas necesidades.
- Casi 200 millones de viajeros pasan actualmente por los aeropuertos españoles, y a muchos de ellos les sería muy útil disponer de una herramienta capaz de guiarles a través de ellos de forma rápida y sencilla gracias a los mapas generados por ANVM.
- Más de 900 hospitales dan servicio a pacientes españoles en la actualidad. ANVM satiface necesidades tanto de los pacientes que necesitan encontrar una sección o una consulta concreta dentro del mapa del hospital como de los visitantes que desean ver y encontrar rápidamente a sus familiares.
- Centros culturales y grandes museos pueden beneficiarse de una virtualización interactiva para mostrar una previsualización de una nueva exposición atrayendo a más público.

Como se puede observar el mercado al que está enfocado ANVM - Virtual Mapping es muy numeroso y trata de ofrecer una funcionalidad que hasta ahora ha sido poco explotada comercialmente, demostrando así su potencial económico.

3. Alternativas y competidores

Existen varias empresas que se dedican a la virtualización de todo tipo de espacios tanto exteriores como interiores, sin embargo, ninguna cumple las necesidades a las que desea dar solución ANVM, y esto es por una parte ofrecer una manera sencilla de obtener un plano o mapa de un espacio interior, y por otra proporcionar al usuario de un smartphone una forma de orientarse dentro de ese espacio. A continuación se van a enumerar y comentar algunas de las alternativas al producto que se han encontrado en el mercado y que se han identificado como posibles competidores dado que estas empresas realizan proyectos en el mismo campo en el que se mueve este subproyecto de ANVM.

3.1. Viametris

Se trata de una empresa francesa dedicada a la producción de software y hardware en el campo de la reconstrucción espacial. Dispone de diversos productos en el mercado capaces de realizar reconstrucciones en tres dimensiones de diferentes espacios interiores con un detalle bastante bueno. Ofrecen también soluciones para recoger estos datos de reconstrucción con sus propios dispositivos y vehículos [5].

Dentro de los proyectos que desarrolla el más similar a ANVM es el llamado I-MMS (Indoor Mobile Mapping System) que ofrece un dispositivo (ver figura 5) capaz de realizar una reconstrucción o SLAM de un espacio interior. Se trata de una adaptación

de otro proyecto de la empresa llamado MMS que realiza este tipo de reconstrucciones pero en lugares exteriores. La empresa presenta la alternativa I-MMS como una más económica y simplificada para el escáner de un espacio interior.

Figura 5: Dispositivo I-MMS de Viametris

- **Ventajas**
  - Realiza una reconstrucción virtual en tres dimensiones sobre espacios tanto interiores como exteriores.
  - La calidad de la reconstrucción tiene bastante buena calidad.
  - Solución muy profesional a la virtualización de espacios.
- **Inconvenientes**
  - No dispone de un sistema para generación de mapas 2D a partir de las virtualizaciones.
  - A pesar de que se ha simplificado el dispositivo para escanear interiores y se han reducido costes sobre él, sigue siendo una alternativa más cara para realizar virtualizaciones.

3.2. Matterport

Se trata de una empresa estadounidense de reciente creación y que desarolla tanto software como hardware para la reconstrucción en 3D. Esta empresa propone la utilización de unas cámaras especiales (diseñadas por ellos mismos) para que el usuario reconstruya él mismo el espacio interior que desee. Los pasos a seguir para realizar una reconstrucción del lugar deseado se pueden consultar en su página web [6] y son los siguientes.

En primer lugar, situar las cámaras en varios puntos diferentes del espacio que se quiere reconstruir (ver figura 6). En segundo lugar, subir a la nube ofrecida por esta empresa

los datos recogidos por las cámaras que se han situado, para que se procesen los datos y se realice la reconstrucción tridimensional del espacio. Por último se puede consultar a través de un enlace la reconstrucción llevada a cabo.

Figura 6: Posiciones de las cámaras para la reconstrucción con Matterport

- **Ventajas**
  - Realiza una reconstrucción virtual en tres dimensiones sobre espacios interiores.
  - La calidad de la reconstrucción tiene bastante buena calidad, y es muy atractiva visualmente.
  - Solución muy sencilla para realizar una reconstrucción de un espacio interior propio.
- **Inconvenientes**
  - No dispone de un sistema para generación de mapas 2D a partir de las virtualizaciones.
  - La utilización de las cámaras puede ser un proceso pesado o molesto para el usuario, ya que lo tiene que realizar por sí mismo.
  - A pesar de que se ha simplificado el dispositivo para escanear interiores y se han reducido costes sobre él, sigue siendo una alternativa más cara para realizar virtualizaciones.

3.3. Google

El gigante americano de la informática moderna cuenta con un gran número de aplicaciones y servicios propios dedicados a la navegación virtual en interiores y exteriores. Vamos a profundizar sobre sus herramientas más populares en este campo.

Google Maps

Se trata de un servicio online ofrecido por Google que muestra al usuario imágenes de mapas obtenidos por satélite. A dia de hoy, Google Maps tiene entre 500 y 1000 millones de descargas en la plataforma Android [7]. Además cuenta con casi 2 millones y medio de valoraciones de usuarios que sitúan su calificación en un 4.4 sobre 5. En la figura 7 podemos ver una captura de esta aplicación.

- **Ventajas**
  - API abierta que proporciona una gran cantidad de servicios de localización a páginas web y aplicaciones para móviles.
  - Gran precisión en exteriores.
  - Existen muchas herramientas integradas de Google que añaden multitud de opciones como Google Ride Finder, Google Sites y Google Traffic entre otros.
  - Cuenta con diferentes capas de visualización para los mapas.
  - Se puede utilizar en casi todos los navegadores y tiene aplicación propia para Android y iOS.
- **Inconvenientes**
  - Es necesario conexión a Internet.
  - En ciudades no demasiado pobladas el nivel de detalle es bajo.
  - Algunas imágenes no son actuales.
  - Invade la privacidad de las personas.
  - No presenta ninguna solución a la visualización de interiores.

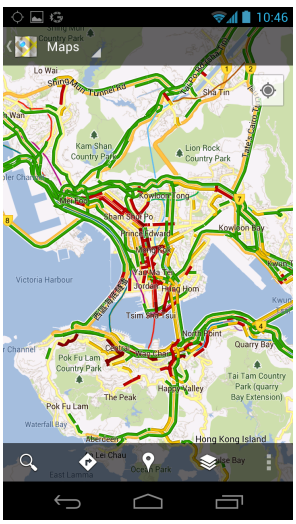

Figura 7: Captura de la aplicación Google Maps

Google Maps para interiores

En Marzo de 2013 esta empresa decide ampliar su abanico de mapas con un conjunto de mapas de ciertos espacios interiores. Se trata de algo novedoso similar a lo que pretende ANVM y que hasta el momento Google había decidido no realizar. De momento se dispone de mapas de unos pocos lugares interiores, pero esta empresa pretende que se sumen poco a poco más espacios interiores al elenco de mapas.

Incluye mapas de lugares públicos de interés como centros comerciales, tiendas, o algunos edificios públicos. Se trata de una iniciativa que continúa en desarrollo ya que acaba de ser presentada hace relativamente poco tiempo. Son mapas sencillos que pretenden ofrecer al usuario una vista rápida y clara del lugar, sin presentar demasiados detalles sobre el recinto, a excepción de algunos puntos que pueden resultar de interés para el usuario como restaurantes o puntos clave del lugar.

En la figura 8 se puede observar la presentación de los mapas que ofrece Google. En la pantalla de la izquierda se muestra el mapa del sitio según google maps original. En la derecha se muestra el mapa del mismo sitio, pero se puede observar cómo se detallan pasillos, tiendas, y elementos del lugar. Esta versión de mapas está disponible sólo para dispositivos con Android instalado, es decir que no se puede consultar desde la web, ni desde otros dispositivos que corran otro sistema operativo como iOS. De momento Google no se ha pronunciado al respecto de si va a sacar esta versión de sus mapas para el resto de dispositivos.

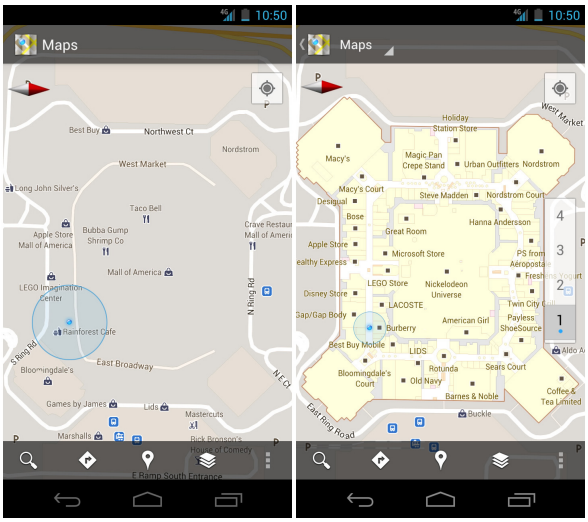

Figura 8: Google indoor maps

- **Ventajas**
  - Se trata de un mapa interior sencillo e intuitivo, similar al que ofrece ANVM.
  - Gran soporte al tratarse de una empresa puntera en el sector.
  - Permite la interacción con amigos que tengan otro dispositivo Android dentro del recinto (buscarlos), mediante el sistema de Google Latitude.

- **Inconvenientes**

  - Está disponible tan sólo para dispositivos que soporten Android, lo que deja fuera una gran franja del mercado, ya que a pesar de que Android es utilizado por aproximadamente un 55.95 % de los usuarios móviles, se deja a la otra mitad sin cobertura, puesto que ni siquiera están disponibles desde un navegador web.

  - Dispone de momento de muy pocos mapas interiores en España.

  - Herramientas para conseguir las virtualizaciones realmente caras, aunque son gratuitas para el usuario.

Google Street View

Complemento de Google Maps utilizado para visualizar los entornos virtualizados. Durante la navegación con Google Maps el usuario puede solicitar la visualización Street View para obtener una panorámica del lugar que pida. Utilizan un sistema en el cual dividen las escenas en nodos. El usuario navega a través de estos nodos bien desde su PC o bien desde su dispositivo portátil. Al igual que Google Maps, el complemento Google Street View cuenta con entre 500 y 1000 millones de instalaciones en Android, con una nota media de 4.4 sobre 5 habiéndose registrado en torno al medio millón de valoraciones. En la figura 9 podemos ver una captura en la que aparecen la interfaz de Google Maps y la de Google Street View en un smartphone con sistema operativo Android.

- **Ventajas**

  - Gran base de datos de ciudades enteras a través de las cuales el usuario puede navegar como si estuviera realmente caminando por una calle en concreto.

  - Gran soporte al tratarse de una empresa puntera en el sector.

  - Navegación a través de fotos reales adaptadas que dan un gran cantidad de realismo a la reconstrucción.

- **Inconvenientes**

  - Al ser un complemento de Google Maps, si el usuario quiere conocer su posición necesita señal GPS. Por ello, en interiores no es apropiado.

  - Necesita de conexión a Internet para descargarse los lugares por los que se va a navegar.

  - Herramientas para conseguir las virtualizaciones realmente caras.

  - Invade la privacidad de las personas.

Figura 9: Capturas sobre Google Maps y Google Street View

Google Art Project

Se trata de una plataforma online desarrollada por Google a través de la cual el usuario puede tener acceso a imágenes en alta resolución de las obras de arte de una gran cantidad de museos a nivel mundial. El usuario tiene la capacidad de navegar por las galerías de los museos como si se tratase de la herramienta Google Street View comentada anteriormente. En la figura 10 podemos ver una captura sobre esta herramienta web [8].

Este tipo de virtualización no es similar a la de este proyecto ya que no es una reconstrucción del espacio, sino simplemente se intenta cubrir el mismo con una serie de imágenes panorámicas. No obstante, ofrece una funcionalidad similar al uso de panoramas en ANVM que aporta una buena visión del espacio interior.

- **Ventajas**
  - Gran calidad de las imágenes.
  - Acerca el arte a los usuarios sin que se tengan que desplazar.
- **Inconvenientes**
  - Coste elevado de la virtualización debido a las herramientas empleadas para ello.
  - No genera mapas del recinto.
  - A día de hoy, no cuentan con una aplicación para dispositivos portátiles.
  - Únicamente están disponibles algunos museos.

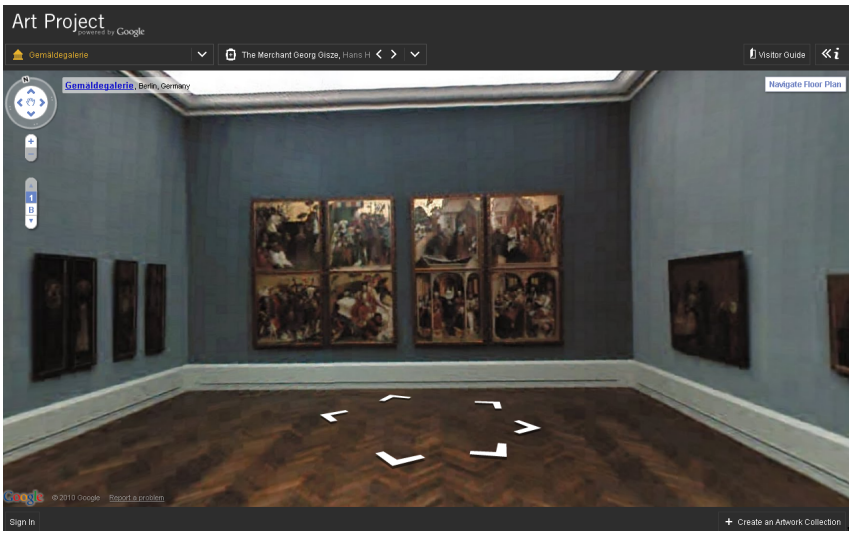

Figura 10: Captura sobre Google Art Project

4. Visión general del producto

En esta sección obtendremos una visión general de ANVM. Se empieza por una perspectiva del producto donde se comentarán los dos proyectos paralelos que forman la suite de herramientas ANVM. Después se enumeran las capacidades del producto y tras ello se pasa a hablar sobre la licencia, las suposiciones y las dependencias de ANVM.

4.1. Perspectiva del producto

Como se comentó en anteriores secciones, este proyecto es una parte de un proyecto conjunto denominado ANVM, siglas de Augmented Navigation and Virtual Mapping. Su desarrollo ha sido dividido en dos grandes partes que han sido llevadas a cabo en paralelo por dos grupos de Sistemas Informáticos de la Universidad Complutense de Madrid (UCM) en colaboración con el grupo de investigación G-Tec y profesores del departamento de DACYA de la UCM. Dichos subproyectos son:

- **ANVM - Virtual Mapping**: Esta primera parte hace referencia a las herramientas y desarrollos necesarios para llevar a cabo la virtualización de los interiores. Con la ayuda del dispositivo Kinect de Microsoft y empleando la información que proporciona un robot construido por los propios alumnos, es capaz de virtualizar en 2D y 3D cualquier entorno interior. Esta virtualización es la que más tarde emplea una aplicación desarrollada en Java, de la que hablaremos más adelante, para detallar y construir el mapa en dos dimensiones con toda la información necesaria. Por último, esta información y este mapa en 2D son los que la aplicación de móvil utiliza para mostrar al usuario final durante la navegación y para el cálculo de rutas.

  Este subproyecto ha sido desarrollado por los alumnos de la UCM, Ricardo Pragnell Valentín y Alejandro Peñalver Santorio y dirigido por los profesores Juan Carlos Fabero y Guadalupe Miñana.

- **ANVM - Mobile App**: Esta otra parte de ANVM engloba todo el ámbito de desarrollo de la aplicación para móviles Android y la aplicación de decorado de mapas desarrollada en Java. Este subproyecto es el encargado de utilizar el mapa que proporciona el proyecto ANVM-Virtual Mapping para su posterior decoración y acotación. Asimismo se encarga de presentar de forma intuitiva para el usuario el mapa dentro de la aplicación móvil, permitiéndole también localizarse dentro del mismo.

  Este subproyecto ha sido desarrollado por los alumnos de la UCM, Miguel Gutiérrez García-Cuevas y Víctor Ortiz García y dirigido por los profesores Luis Garmendia y Victoria López.

El nexo de unión entre ambos proyectos es la aplicación desarrollada en Java ANVM-Map Editor. Este software fue desarrollado por los cuatro miembros del equipo, y se explica como subproyecto en la memoria [9]. Esta aplicación toma la virtualización en 2D del entorno en una vista cenital, a modo de plano. Sobre esta imagen se añadirán los iconos correspondientes a los puntos de interés que se quieran remarcar. Aparte del apartado visual, también habrá que situar los puntos de interés sobre el mapa y crear una red o grafo sobre el cual se calcularán las rutas cuando el usuario solicite el camino óptimo para llegar a un lugar. En el diagrama de la figura 11 podemos ver ilustrada esta distribución.

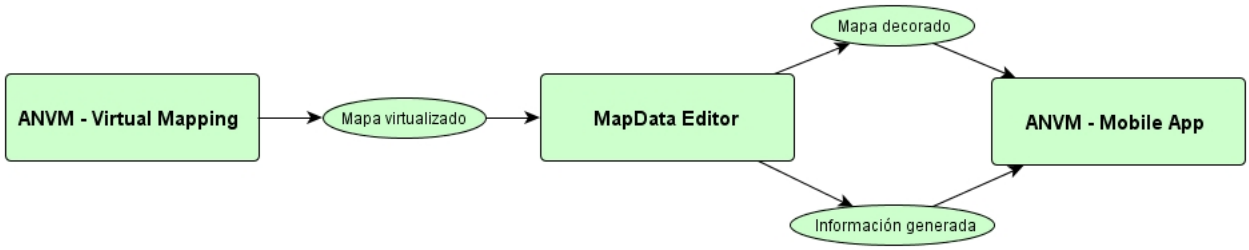

Figura 11: Overview ANVM

4.2. Sumario de capacidades

El cuadro 6 muestra una tabla sobre las capacidades del producto relacionadas con los beneficios que aporta a los usuarios.

Beneficios para el usuario	Capacidades del producto
Disponer de un mapa o plano de algún lugar interior desconocido para el usuario.	Gracias a la virtualización llevada a cabo por el equipo de desarrollo de ANVM se pueden obtener mapas de casi cualquier lugar.
Las empresas y PYMES pueden disponer de una altenativa para virtualizar sus centros y de esta manera sus clientes encuentren más confortable el recorrido por el complejo.	Apoyándose en un pequeño robot de coste muy económico y procesando los datos que recoge, ANVM puede fácilmente reconstruir estos complejos.
Las empresas y PYMES pueden también publicitar algunos espacios u ofertas que deseen.	Gracias a la realidad aumentada, a través de la cámara del teléfono móvil se puede mostrar información de la imagen que se está visualizando.
Disponer de una lista de lugares y puntos de interés en la que se pueden realizar búsquedas.	La aplicación de móvil de ANVM da la posibilidad de buscar lugares dentro de una pequeña base de datos.
Poder optimizar el tiempo para desplazarse de un lugar a otro del centro virtualizado.	ANVM tiene la capacidad de calcular rutas óptimas en distancia entre distintos puntos del mapa que seleccione el usuario.
Poder localizarse en cualquier momento dentro del mapa del complejo.	ANVM da la posibilidad de utilizar códigos QR para poder localizarse dentro del mapa del lugar.
Poder obtener información de cada punto de interés e incluso una vista panorámica del entorno para familiarizarse previamente antes de acudir a él.	Cada punto de interés de la base de datos cuenta con una ficha en la que se encuentra una descripción del propio punto y las opciones para desplazarse hacia allí, verlo situado en el mapa 2D o ver una vista 3D.
Poder disfrutar de la experiencia de uso de ANVM en varios idiomas.	ANVM está disponible en distintos idiomas para tratar de llegar asi a una comunidad de usuarios mayor.

Tabla 6: Sumario de capacidades

5. Características del producto

Las principales características de las que puede presumir ANVM son las siguientes:

- **Virtualización de lugares cerrados en forma de mapas.** Como se explica en la memoria del proyecto paralelo (ANVM - Virtual Mapping), con la ayuda de un pequeño robot construido por los propios desarrolladores y la utilización de cierto software especializado, se obtienen reconstrucciones en dos dimensiones de los lugares deseados.

- **Navegación por un mapa.** Se puede consultar el mapa en la pantalla del teléfono móvil y navegar a través de él realizando acciones de zoom para obtener más detalle sobre una parte concreta. Además el mapa está dotado de iconos y letreros que facilitan su interpretación por parte del usuario.
- **Puntos de interés.** Existe una base de datos de puntos de interés. Estos puntos pueden ser consultados a través de una lista en la que aparecen todos ellos ordenados alfabéticamente o realizando búsquedas de alguno en concreto que se solicite.
- **Localización exacta dentro del mapa.** Utilizando la cámara del smartphone se escanean una serie de códigos QR situados a lo largo de las instalaciones que están asociados a una localización concreta. De esta manera se puede situar el usuario dentro del mapa y considerar de un vistazo los puntos de interés más cercanos.
- **Rutas óptimas entre puntos del mapa**. Una vez seleccionado cierto punto de interés se puede elegir la opción de ir hasta allí desde la posición actual. De otro modo, una vez localizados en el mapa, podemos buscar el lugar al que queremos ir para recibir la ruta más rápida. De cualquier manera, el usuario recibe siempre la ruta optimizada en distancia para dirigirse de un punto.
- **Interfaz gráfica seria y con acabados profesionales.** La interfaz de la aplicación Android de ANVM resulta impactante y cuenta con una disposición de colores, fondos, fuentes e imágenes que convierten la experiencia de uso en algo realmente sencillo e intuitivo a la par que vistoso para cualquier tipo de usuario.
- **Visualización de la aplicación en distintos idiomas.** Con el fin de poder llegar a una mayor comunidad de usuarios, ANVM - Mobile App cuenta con la ventaja de poder cambiar de idioma pudiendo seleccionar entre inglés, español y francés.
- **Visualización del lugar a través de panoramas cilíndricos.** Gracias a esta funcionalidad el usuario es capaz de familiarizarse con el entorno virtualizado a través de una navegación entre los diferentes panoramas cilíndricos que lo conforman.
- **Guía de ayuda en situaciones de emergencia.** Dada una situación de emergencia, la aplicación permite al usuario, mediante un acceso directo a esta característica, la posibilidad de localizarse en el mapa e indicarle la ruta óptima de evacuación a la salida de emergencia más próxima.

Capítulo 3

Investigación

En esta sección se describirá todo el proceso de investigación del proyecto. Debido a que el proyecto está enfocado en resolver un problema que se encuentra actualmente en pleno desarrollo como es la localización en interiores, ha sido necesaria mucha investigación en diversos campos. Según la idea inicial del proyecto, que era intentar localizarse en un espacio cerrado, se investigaron técnicas que resuelven este tipo de problemas, todas enfocadas en localización mediante imágenes, o conjuntos de imágenes. Relativo a este tema se encuentra abundante información debido al incremento de actividad que ha tenido lugar en los últimos años sobre sistemas reconocedores de imágenes que permitan este tipo de técnicas. A su vez, aunque haya mucha información, la mayoría de los proyectos de investigación sobre estos temas no están terminados, o dejan cuestiones sin resolver, lo que dificulta en gran medida su comprensión y aplicación a este proyecto.

Sensor Kinect™ de Microsoft™

Un ejemplo claro de estos avances es el sensor Kinect [10] de la compañía Microsoft (Ver figura 12), que fue sacado al mercado el 4 de Noviembre de 2010. A pesar de que existían antes otros dispositivos similares, éste fue el primero en producirse de forma masiva, y a un precio razonable, lo que hizo posible que diversos investigadores aprovecharan la tecnología y se comenzara el desarrollo de programas que se benefician de este sensor. Como se comentará más adelante, el dispositivo Kinect ofrece unas posibilidades que hasta hace poco no eran posibles, y que tienen multitud de aplicaciones. No obstante en este proyecto no se aprovecha toda la funcionalidad que permite el sensor ya que se simplifica las capturas en 3D que realiza este dispositivo a capturas en 2D y por tanto se utilizará de una forma más básica a como se concibió el proyecto inicialmente. Las dificultades encontradas durante la fase de investigación motivaron que ésta tomara otro camino que nos aleja de la funcionalidad en 3D del sensor Kinect, y nos lleven a un uso más “básico” del Kinect como sensor láser 2D.

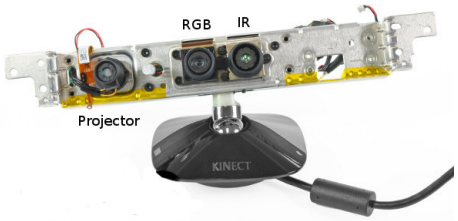

Figura 12: Elementos Principales Dispositivo Kinect

El objetivo principal de Kinect en este proyecto es el de proporcionar nubes de puntos de una determinada zona o área. Una nube de puntos no es más que un gran conjunto de puntos en 3D, es decir, con 3 coordenadas X, Y, Z. Para la obtención de dichas nubes, Kinect dispone de una cámara y un sensor de infrarrojos y un emisor láser. La cámara

permite visualizar el entorno en 2D (coordenadas X, Y) y gracias a la emisión de un patrón fijo de haces infrarrojos (ver figura 13) se puede determinar la profundidad de cada uno de los puntos de la nube (la coordenada Z) lo que nos da las tres dimensiones.

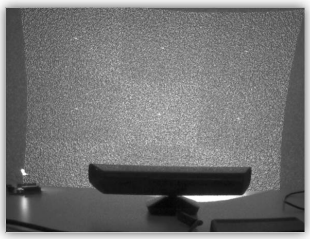

Figura 13: Patrón del haz infrarojo.

Para el uso de este dispositivo se examinaron varias librerías que permiten que el control del Kinect sea más sencillo. Cada librería ofrece una funcionalidad básica similar (obtener y almacenar datos del dispositivo) pero divergen en las opciones de tratamiento de los datos (filtros, visualización, segmentación, etc). Cabe destacar que la mayoría son librerías recientes, proyectos que no han sido acabados ya que como se ha mencionado antes, éste es un tema en pleno desarrollo. Por esto mucha funcionalidad, a pesar de estar investigada teóricamente no ha sido implementada (o no con un rendimiento óptimo). En la sección 3 se presentarán las librerías que se investigaron, se explicarán brevemente sus ventajas y desventajas, y se explicarán por qué se utilizaron, o por qué no.

1. SLAM: Simultaneous localization and mapping

SLAM es el objetivo principal que se pretende conseguir en este proyecto y consiste en esencia en realizar una reconstrucción o virtualización de un cierto espacio (en este caso interiores). El método SLAM no sólo pretende obtener la reconstrucción de un área desconocida (ningún dato previo), sino también conocer la posición exacta del sensor dentro de la reconstrución. Como sus siglas indican, el mapa se va generando simultáneamente a la localización a medida que se va avanzando por el espacio que se está reconstruyendo. Existen diferentes técnicas para llevar a cabo este cometido y en este apartado vamos a explicar los aspectos básicos de los tipos de SLAM que se han investigado y que han tenido su influencia en el proyecto, para que después, a la hora de referirnos a ellos en la sección 3 de Librerías y proyectos investigados quede todo totalmente claro.

1.1. 3D Slam

Se trata del primer objetivo de este proyecto, y no es otro que realizar una reconstrucción 3D de un espacio. Generalmente la forma de realizar este tipo de virtualización es mediante la utilización de nubes de puntos en 3D. Aunque existen otros métodos

diferentes, la mayoría están basados en las nubes de puntos. La idea básica es ir reconociendo el espacio que se quiere reconstruir con un dispositivo capaz de captar nubes de puntos y a partir de ahí unir progresivamente cada nube de puntos con la siguiente de manera que se forma una única nube global y consistente que representa el espacio interior virtualizado. Este proceso de combinado es conocido como *registration* y forma parte de la cadena de tareas de SLAM (Ver figura 14).

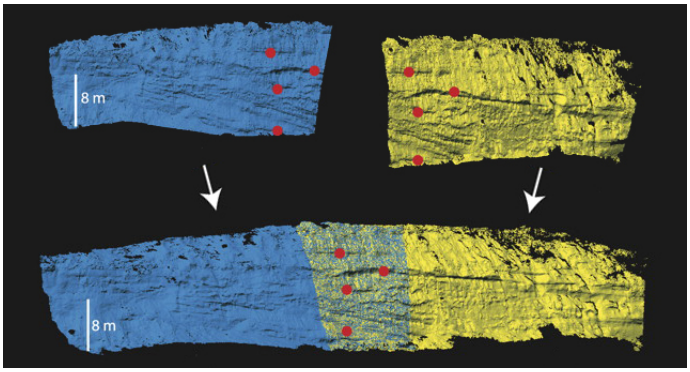

Figura 14: Fusión de dos nubes de puntos 3D

En la figura 15 se puede observar una reconstrucción mediante nubes de puntos 3D. A continuación se presentarán los diferentes tipos de 3D SLAM que se han investigado en este proyecto.

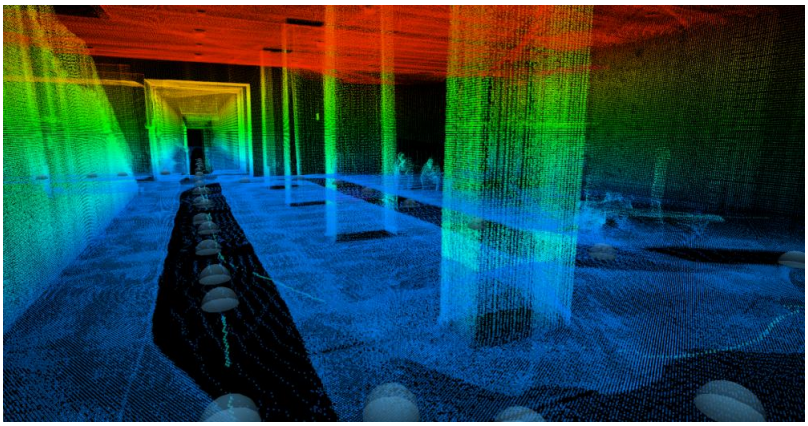

Figura 15: 3D SLAM mediante nubes de puntos

3D SLAM usando Kinect

Se pretendía realizar una SLAM 3D mediante nubes de puntos obtenidas con Kinect, que permitiera después localizar al usuario de la aplicación mediante la localización basada en imágenes (ver sección 2). La idea era llevar el Kinect por todo el espacio interior construyendo el mapa 3D a partir de la concatenación de las nubes de puntos que se iban generando. Para realizar este encadenamiento de las nubes de puntos se pueden utilizar diferentes algoritmos, siendo el ICP (Iterative Closest Point) el más conocido de todos.

Mediante este algoritmo se superpone cada par de nubes y se analiza iterativamente los puntos resultantes más próximos entre sí (Closest Point) hasta que se encuentra una buena aproximación con un error pequeño y se puede decir que ambos puntos de las dos nubes son "el mismo". De esta manera podemos obtener finalmente una nube de puntos que represente todo el espacio que hemos ido reconstruyendo. Este método se puede implementar con la ayuda de varias librerías las cuales expondremos más adelante. Sin embargo esta técnica no ha sido implementada actualmente en este proyecto ya que el tamaño de las nubes de puntos, aún filtradas y comprimidas, ocupan una elevada cantidad de espacio en relación con el espacio reconstruido haciendo complicado su tratamiento y almacenamiento. No se descarta poder utilizarla en un futuro dado que puede ser útil para la localización basada en imágenes (ver sección 2).

3D SLAM mediante modelos

Existen también otras alternativas que ofrecen un resultado visiblemente más atractivo para las reconstrucciones en 3D, como es la reconstrucción mediante modelos 3D. En realidad esta técnica se basa en la reconstrucción mediante nubes de puntos que se ha explicado anteriormente, y tras haber obtenido una nube de puntos de lo que se está visualizando, se aplican ciertos algoritmos para transformar esos conjuntos de puntos 3D en polígonos que definan los modelos de los objetos tridimensionales. De esta manera se obtienen resultados más agradables a la vista que con las nubes de puntos. Sin embargo, se trata de una técnica más difícil de implementar, y que no siempre produce el efecto deseado, es decir, los polígonos que se forman pueden no ser los que deberían y se produciría un efecto inesperado. Además requieren una capacidad y velocidad de procesamiento de datos mucho más elevadas. A continuación en la figura 16 se muestra un ejemplo de los objetos 3D que se pueden virtualizar utilizando esta técnica.

Figura 16: 3D SLAM mediante modelos

1.2. 2D Slam

En este caso, la reconstrucción del espacio tiene como objetivo la obtención de un mapa 2D. En el caso de un espacio interior, como una oficina por ejemplo, se correspondería con un plano de dicha oficina. Actualmente, ésta es la técnica que se ha utilizado para reconstruir espacios en el proyecto ANVM. Para la obtención de este tipo de mapas se

suelen utilizar láseres que exploren el espacio en un solo plano generando una imagen (el mapa 2D). Sin embargo en este proyecto se propone el uso del dispositivo Kinect como una alternativa más económica y sencilla para realizar esta tarea.

Existen diferentes métodos para realizar este tipo de virtualizaciones y para este proyecto se investigaron algunos como por ejemplo el RBPF slam, el ICP slam, etc. que serán explicados más adelante junto con las librerías que permiten su implementación. Concretamente se estudiaron dentro de la librería ROS (ver sección3.3) y MRPT (ver sección 3.5). El resultado obtenido en el 2D SLAM no es tan atractivo visualmente hablando, pero cumple muy bien su función de proporcionar un mapa sobre el que trabajar en ANVM-Mobile app [9]. Se puede observar un ejemplo de 2D SLAM en la figura 26 de la sección 3.3.

2. Localización basada en imágenes

Este proyecto intentó desde un principio aportar una solución a la localización basada en imágenes, como se ha descrito anteriormente. Esta línea de investigación está generando mucha actividad en la actualidad, y se realizó una investigación exhaustiva sobre el tema para poder plantear una solución a la misma. La aplicación más directa de esta técnica, y la que se quería aplicar en este proyecto, es la ubicación de una imagen en un conjunto muy grande de imágenes. Imaginemos que un usuario se encuentra dentro de un espacio interior que ha sido reconstruido previamente por nosotros. Si dicho usuario quisiera localizar el punto exacto en el que se encuentra, lo único que tendría que hacer es sacar una foto desde donde se encuentra. Esta técnica permitiría emplazar la foto en el subconjunto de imágenes recogidas anteriormente del espacio interior (reconstrucción) y mediante unos cálculos geométricos indicarnos en qué punto de la reconstrucción nos encontramos.

La localización basada en imágenes es un proceso bastante complejo desde el punto de vista de computacional, ya que a pesar de existir algoritmos para solucionar la mayoría de los problemas que plantea, muchos son poco eficientes, y no pueden ser implementados en dispositivos con una capacidad de cómputo relativamente baja (smartphones o tablets) o ni siquiera en una estación de trabajo potente. A continuación se va a explicar más detalladamente en qué consiste este proceso, cuáles son las claves y los principales inconvenientes.

2.1. Keypoints

Uno de los pilares de la técnica de la localización basada en imágenes son los puntos característicos o *keypoints*. Éstos son puntos de la imagen que aportan cierta información significativa. Consideremos como ejemplo el que se muestra en la figura 17. Es una foto capturada con una cámara fotográfica estándar. En ella podemos apreciar los keypoints extraídos marcados mediante cruces amarillas. Además vemos que estos puntos interesantes se encuentran en secciones de la imagen que aportan una cierta diferencia con respecto a lo que tienen alrededor (habitualmente en las esquinas, marcos de ventanas y la frontera entre el tejado y la fachada). Si tenemos una mesa en la imagen,

las esquinas de la mesa serían keypoints porque representan a un objeto, lo delimitan, sin embargo, todo el tablero que conforma la mesa no dispone de puntos de interés, porque son todos iguales a su alrededor, es decir, no aportan rasgos distintivos y por ello carecen de información útil desde el punto de vista de la reconstrucción.

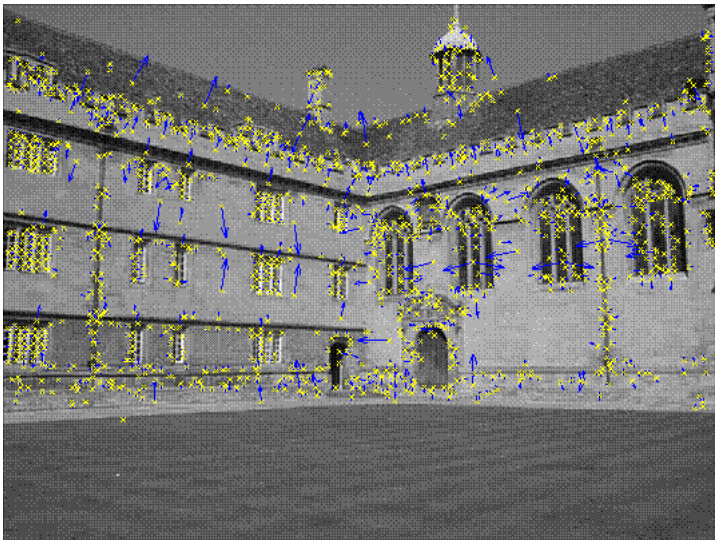

Figura 17: Keypoints

Los keypoints son la base para realizar el emparejado o matching de una imagen. Es común que en una imagen tomada con una cámara de media a alta resolución se obtenga una cantidad elevada de píxeles, y si intentáramos buscar para cada uno su correspondiente en el conjunto de imágenes la búsqueda sería inabordable. Gracias a los keypoints la búsqueda se reduce y además proporciona mayor capacidad de acierto. Se reduce puesto que evidentemente son menos puntos con los que hay que comparar y es más eficaz porque estamos comparando sólo los puntos que aportan verdadero significado a la imagen, dejando los puntos más comunes que no aportan información relevante y que pueden llevar a confusión.

Para una mayor facilidad a la hora de procesar estos puntos se usan los llamados descriptores. Cada descriptor se identifica con un keypoint y trata de representar a dicho punto clave de forma única. Existen diferentes tipos de descriptores, así como diferentes tipos de keypoints. Algunos guardan información sobre el color del píxel, otros sobre el color del conjunto de píxeles que rodean al punto... Habitualmente se confunden descriptores y keypoints, dado que son esencialmente la misma cosa: keypoint se refiere al punto en sí mismo y el descriptor a su representación. La principal medida de calidad de los keypoints es su robustez, es decir, si el punto clave es capaz de resistir distorsiones, desenfocado, ruido y cambios de ángulo. También existen herramientas que nos permiten calcular los keypoints de una imagen (y sus descriptores) aunque no todas identifiquen los mismos puntos.

A continuación se explican los diferentes tipos de keypoints investigados, así como sus descriptores, ventajas e inconvenientes.

Descriptores SIFT

Los descriptores SIFT (Scale-Invariant Feature Transform) están patentados en EEUU, son propiedad de la University of British Columbia y son sin duda los más utilizados. Esto es debido a que son descriptores relativamente simples, robustos y rápidos de calcular. David Lowe publicó este algoritmo en el año 1999 se utiliza en muchos campos relacionados con la visión por computador como reconocimiento de objetos, robótica y reconocimiento de gestos.

La robustez de un descriptor determina la capacidad del descriptor de aguantar distorsiones, ángulo, cambios de iluminación, escalado, etc en la imagen y continuar describiendo el punto clave.

Cada descriptor contiene información sobre una cuadrícula que engloba a los 255 píxeles que rodean al keypoint. Esta cuadrícula se divide a su vez en 16 cuadrículas de 4x4 píxeles cada una y a cada cuadrícula de 4x4 se le asocia un histograma. Cada histograma está definido por 8 contenedores (bins) que describen el valor en una escala de grises de los píxeles de ese histograma. Cada contenedor representa un rango dentro de la escala de grises es decir, el contenedor nº 1 correspondería al rango 0-31, el contenedor nº2 al rango 32-63 y así hasta el contenedor nº8 que corresponde al rango 224-255 dentro de la escala de grises. De esta manera, si un píxel tiene el valor (en escala de grises) 125 le correspondería el bin nº4. Al final de este proceso cada histograma representa un vector de 8 elementos (cada uno correspondiente a un contenedor) que contiene en cada entrada el número de píxeles que han ido a parar a ese contenedor. En conclusión tenemos 8 contenedor por cada histograma, 16 histogramas por cada descriptor, un total de 8 * 16 = 128 elementos almacenados en un vector al que llamamos descriptor SIFT del keypoint.

David Lowe propone también una implementación de dicho algoritmo en su página web que permite visualizar la extracción de los SIFT keypoints de una imagen en formato .pgm. A continuación se muestra un ejemplo de los puntos de interés extraídos con esta herramienta (Figura 18).

Figura 18: SIFT keypoints obtenidos con la herramienta de David Lowe

Descriptores SURF

Los descriptores SURF (Speeded Up Robust Features) presentados por Herbert Bay en 2006 están parcialmente basados en los descriptores SIFT, pero son varias veces más rápidos y robustos a transformaciones en la imagen. Una de las principales diferencias de estos descriptores es que no utilizan colores para formar el descriptor sino 2D Haar *wavelets* y un uso eficiente de imágenes integrales. Estas *wavelets* u ondículas representan la intensidad luminosa de los puntos cercanos al punto característico del cual se está sacando el descriptor. Para una explicación más detallada consultar la publicación original en el artículo Speeded Up Robust Features Herbert Bay , Tinne Tuytelaars , and Luc Van Gool. En la gráfica puede apreciarse la mejora de los descriptores SURF frente a los SIFT. Sobre el eje X tenemos la precisión con la que hemos sacado el descriptor, y en el eje Y tenemos las correspondencias para dicha precisión. Podemos observar que los descriptores SURF-128 son los que mejor resultado aportan [11].

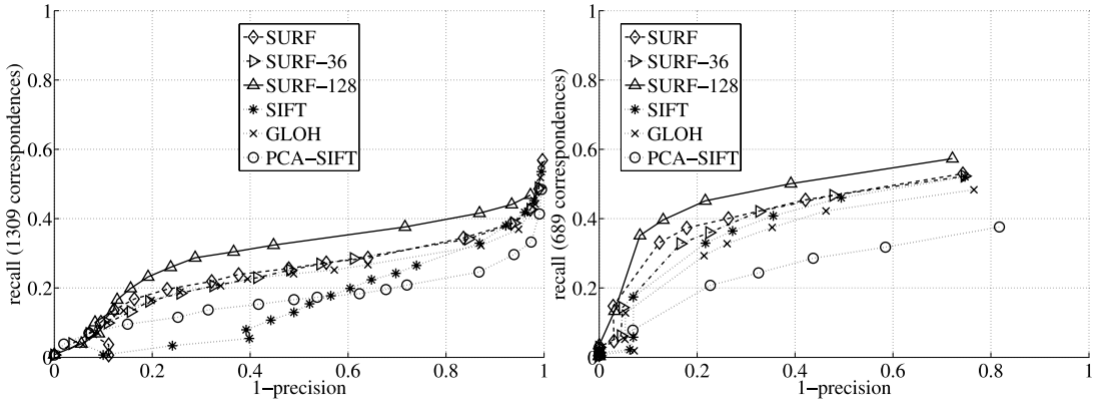

Figura 19: Comparación precisión descriptores SURF vs SIFT

Descriptores FAST

Los descriptores FAST (Features from Accelerated Segment Test) fueron desarrollados por E. Rosten y T. Drummond en 2006. Consisten en una versión más generalizada de los descriptores SUSAN (Smallest Univalue Segment Assimilating Nucleus) que permiten la detección de esquinas y bordes en la imagen. Se puede consultar más información sobre ellos en la publicación original: S. M. Smith and J. M. Brady (May 1997). "SUSAN – a new approach to low level image processing". International Journal of Computer Vision [12].

Ambos descriptores están basados en detección de esquinas. FAST analiza una circunferencia de 16 pixels y su centro o núcleo. A partir de los niveles de brillo de dichos píxeles es posible obtener descriptores robustos (los descriptores de píxeles parecidos son también parecidos, esto es que representan bien al píxel) en un tiempo computacional muy bajo, por tanto son muy indicados para desarrollo en móviles. En la figura 20 podemos ver una captura que ilustra la explicación. La *p* roja representa el píxel del que se va a sacar el descriptor. Para este descriptor se tienen en cuenta los 16 píxeles

recuadrados en rojo más el píxel central *p*. Estamos por tanto cogiendo información de la esquina p y de los píxeles que rodean y definen la esquina.

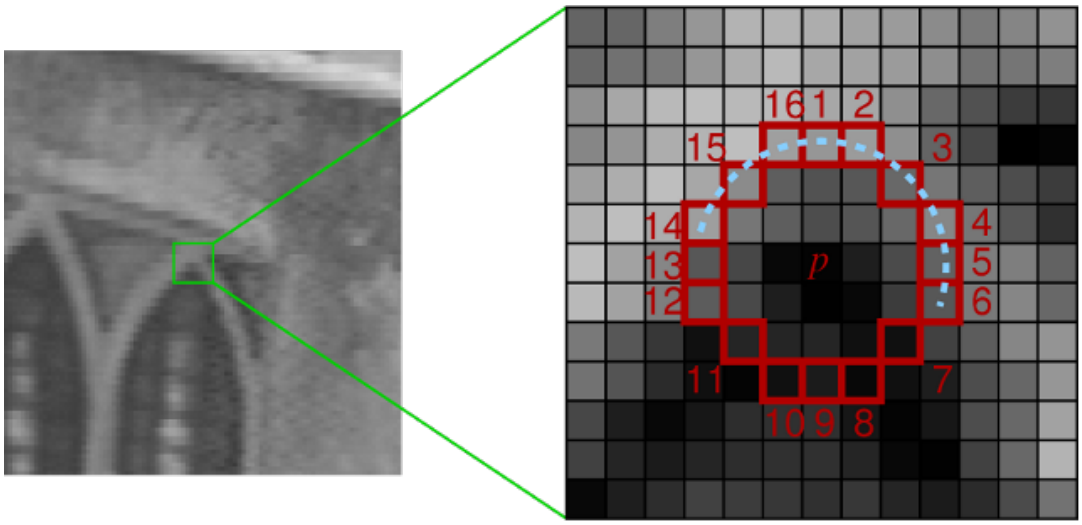

Figura 20: Ampliación descriptor FAST

Conclusiones

Estos 3 tipos de descriptores son sólo algunos de los muchos que existen. Los descriptores SIFT tienen la ventaja de ser sencillos de calcular, y más o menos robustos. Se dispone además de una gran cantidad de aplicaciones que los utilizan y está bastante probada su eficiencia. En cuanto a los descriptores SURF, son parecidos a los SIFT pero ofrecen una mejora en el tiempo de ejecución a la hora de calcularlos y son más robustos. Sin embargo son más actuales y todavía no se disponen muchas implementaciones de ellos. Los descriptores FAST proponen una aproximación que puede ser utilizada en otros descriptores, puesto que llevan una buena detección de esquinas y bordes como puntos de interés.

2.2. Structure-from-Motion

Structure-from-Motion (SfM) hace referencia a un proceso de reconstrucción de la estructura tridimensional de la realidad a través del análisis de información 2D (fotografías). En biología, SfM se utiliza para nombrar al fenómeno por el cual los humanos y otros animales pueden reconstruir una escena 3D de la proyección en 2D sobre su retina. Los seres humanos perciben una gran cantidad de información acerca de la estructura tridimensional de su entorno moviéndose a través de él. Si el observador se mueve y los objetos que le rodean también, se puede extrapolar información visual a lo largo del tiempo que permite a su vez obtener coordenadas globales del entorno.

Para encontrar la correspondencia entre las imágenes, son necesarios puntos característicos como las esquinas de los objetos (puntos en los que el gradiente toma muchas direcciones). Las trayectorias de estos puntos característicos a lo largo del tiempo son usados para reconstruir la escena en 3D y el movimiento de la cámara.

Dentro del ámbito que nosotros estamos estudiando, SfM es necesario para lograr reconstrucciones del entorno en el cual posteriormente llevaremos a cabo el cálculo de la localización. Para ello existen implementaciones que realizan este proceso. Aquí presentamos las diferentes implementacionse de SfM que se probaron.

Bundler SfM by Noah Snaverly

Bundler es un término inglés que hace referencia a la acumulación de información referente a algún tema para la obtención de un cierto resultado [13]. En el tema en el que nos estamos moviendo se refiere a la obtención de información referente a imágenes para poder construir una escena en tres dimensiones con dicha información. Se trata por tanto de un SfM que fue desarrollado por N. Snaverly en C y C++ y utilizado en el proyecto Photo Tourism. Este bundler toma como entrada un conjunto de imágenes, los puntos característicos de cada imagen, y los emparejamientos (matches) de cada imagen. Los emparejamientos de las imágenes se refieren a los puntos que tienen en común un par de imágenes, puntos que se refieren al mismo objeto en la ecscena que se está visualizando. Tanto los puntos característicos como los emparejamientos deben haber sido calculados previamente por algún otro método.

El sistema reconstruye la escena incrementalmente usando como motor base una versión modificada del paquete de Sparse Bundle Adjustment de Lourakis y Argyros. Como salida, aparte de la reconstrucción de la escena en 3D en forma de nube de puntos, también aporta información intermedia, como los descriptores SIFT de los puntos 3D, necesaria para cálculos posteriores. Este bundler se ha ejectutado con éxito en numerosas colecciones de fotos de internet.

En la figura21 podemos ver el resultado de la reconstrucción del Coliseo Romano utilizando este software. Como podemos observar hay una gran variedad de fotos tomadas desde diferentes puntos y ángulos, entre todas capturando la totalidad del Coliseo. Se obtienen resultados mejores cuando las imágenes son tomadas con algo de sentido y no aleatoriamente. Hemos comprobado que se logran grandes resultados en la reconstrucción si se toman fotografías de los mismos objetos de la reconstrucción variando ligeramente el ángulo. Es decir, si se está reconstruyendo una torre por ejemplo, conviene tomar una imagen apuntando a la puerta principal, y otra variando la posición de la cámara unos pasos hacia un lado pero manteniendo parte de la puerta. El solapamiento de zonas entre imagenes es necesario para poder obtener un resultado aceptable. Si además queremos capturar la escena completa debemos asegurar la cobertura de imagenes de toda la zona.

Figura 21: Reconstrucción del Coliseo Romano

VisualSfM by Chanchang Wu

Esta otra alternativa desarrollada por C. Wu, investigador del departamento Computer Science & Engineering de la Universidad de Washington también tiene la posibilidad de llevar a cabo reconstrucciones partiendo de un conjunto de imágenes. Esta herramienta cuenta además con una interfaz desde la cuál poder trabajar [14]. Como entrada, únicamente es necesario cargar las imágenes que van a ser estudiadas. Una de las ventajas que tiene es que también puede realizar una reconstrucción optimizada tomando como entrada una secuenca de fotogramas de vídeo en alta definición.

Como primer paso tras cargar las imágenes, se calculan los SIFT points de las imágenes y las correspondencias entre ellas, y son almacenados en archivos .sift y .mat respectivamente en la propia carpeta donde se encuentran las imágenes, estando disponibles para uso posterior, bien en la propia herramienta o en herramientas de terceros. Con estos descriptores SIFT y emparejamientos de cada imagen se lleva a cabo la reconstrucción progresiva, realizando emparejamientos entre los puntos 2D y 3D. La salida será una reconstrucción de la escena en 3D que aparecía en las fotografías.

Se trata de un software paralelo, mucho más optimizado y eficiente que la alternativa que hemos propuesto anteriormente. No obstante, el formato de la salida no es exactamente el que necesitábamos para la localización.

Conclusión

Estos métodos pueden ser utilizados para generar las escenas en 3D sobre las que luego realizar la localización basada en imágenes. Durante la investigación de este proyecto se utilizó una herramienta llamada Acg_Localizer capaz de localizar a un usuario en una escena 3D a partir de una imagen. De esta herramienta hablamos en la sección 3.2 y se recomienda usar la primera alternativa desarrollada por N. Snaverly que es la que utilizamos nosotros. No obstante, también exploramos las posibilidades de utilizar la segunda alternativa (VisualSfM) para poder extraer los SIFT points y los emparejamientos de las imágenes en menor tiempo. Finalmente decidimos utilizar la primera alternativa (Bundler SfM), como bien recomiendan los desarrolladores de Acg_Localizer, ya que de otro modo requeriría un paso intermedio de conversión de formatos de datos de salida, perdiendo cualquier beneficio de velocidad que aportaba VisualSfM.

2.3. Estimación de la posición

Este es el punto final en la localización basada en imágenes. Si suponemos que ya tenemos nuestra imagen situada dentro la reconstrucción, lo único que nos queda es calcular la posición exacta del usuario que ha tomado la foto dentro del espacio 3D reconstruido. Existen variedad de algoritmos matemáticos para realizar esta operación, y uno de los más utilizados es el conocido como 3 Point Pose Estimation (estimación de la posición por tres puntos).

Estimación de la posición por tres puntos

En el paper "Real-Time Self-Localization from Panoramic Images on Mobile Devices" de Clemens Arth, Manfred Klopschitz, Gerhard Reitmayr y Dieter Schmalstieg de la Universidad de Graz, Austria se propone un modelo de localización basada en imágenes apoyado en esta técnica. Proponen un modelo eficiente y sencillo, para que pueda ser utilizado en dispositivos con baja potencia de cómputo como smartphones.

En primer lugar realizan una reconstrucción del espacio mediante el proceso SfM, explicado anteriormente. Después intentan hallar la posición de una imagen panorámica dentro de ese espacio reconstruido. La razón para usar imágenes panorámicas es que el uso de la técnica de estimación por tres puntos es más sencilla con imágenes panorámicas y resuelve el problema del campo de visión (FOV) limitado de una cámara normal. Dentro de la imagen panorámica que se quiere reconstruir se eligen 3 puntos 2D con sus correspondencias a puntos 3D de la reconstrucción. Se considera la imagen como un cilindro dentro del espacio 3D (dado que es panorámica) y se trazan 3 rayos desde cada uno de los puntos de la imagen cilíndrica (panorámica) a sus correspondencias en 3D. El punto resultante del cruce de los 3 rayos será la posición desde la que se ha tomado la imagen.

En la figura 22 se puede observar un gráfico de este proceso, donde el cilindro sería la imagen panorámica, los puntos $m_i$corresponden a los puntos 2D de la imagen, y los puntos x_i a las correspondencias 3D de la reconstrucción. El punto C es el punto estimado de la posición del observador.

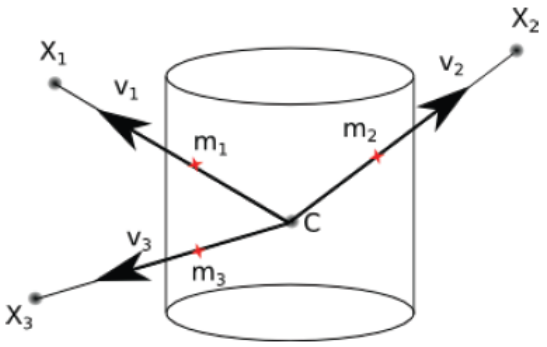

Figura 22: 3 point pose estimation

3. Librerías y Proyectos investigados

La investigación en este proyecto está totalmente ligada a su desarrollo, puesto que la mayoría de los conocimientos necesarios para desarrollarlo no han sido adquiridos durante la carrera, al ser éste un tema muy actual y en desarrollo. Por esta razón se va a intentar explicar con el mayor detalle posible cada uno de los temas investigados, de los cuáles hemos sacado las conclusiones sobre el camino que debería llevar la aplicación. Siempre teniendo en cuenta que es un proyecto fácilmente ampliable y modificable y que, actualmente, la omisión de ciertas técnicas, por no cumplir estrictamente los requisitos o porque se ha estimado que pudieran llevar demasiado tiempo en ser implementadas, puede cambiar a lo largo del tiempo y pueden ser incluidas en el proyecto en un futuro.

3.1. PCL : Point Cloud Library

PCL es una librería muy potente que facilita el manejo de la información que suministra el dispositivo Kinect y fue la primera opción que se contempló para desarrollar el proyecto. Se consideró una buena opción para comenzar, porque se trataba de un librería con licencia BSD y por lo tanto libre para su uso comercial e investigación. También parecía una buena opción porque dispone de mucha funcionalidad en el tratamiento de imágenes (tanto 2D como 3D) y nubes de puntos, mas allá del simple manejo del dispositivo. Cuenta además con bastante apoyo por diversas organizaciones como Nvidia, Google, Toyota y Ocular Robotics, por mencionar algunas, lo que asegura un continuo desarrollo de nuevas funcionalidades así como estabilidad para las ya existentes dentro de la librería. A continuación vamos a explicar un poco más detalladamente lo que ofrece esta librería, y qué cosas fueron útiles o no, para el posterior desarrollo del proyecto.

PCL comenzó su desarrollo en Marzo 2010 por parte de WillowGarage creadores de ROS, una librería muy utilizada en el campo de la robótica. Impulsada por varios code sprints (grupos de desarrolladores) subvencionados por grandes entidades como Google y Nvidia se ha convertido en la mayor librería de captura y tratamiento de nubes de puntos, además de otras funcionalidades. Entre las posibles herramientas que ofrece destacamos OpenNI como controlador para obtener capturas de sensores como LIDAR, Kinect, XTionPRO, etc; numerosos filtros para reducir el número de puntos de la nube; técnicas de registration para unir varias nubes; cálculo de puntos clave *(keypoints)* y descriptores 3D como Narf, Harris, etc; procesos pioneros de segmentación de escenas, reconociendo los objetos particulares contenidos en ella; y reconstrucción poligonal entre otros. Actualmente continúa su desarrollo incorporando nuevas plataformas como OSX y Android, pasando a formar parte de una organización más general, Open Perception, centrada en la misión de proveer a los computadores con medios de percibir el mundo 3D.

Nubes de puntos

La principal funcionalidad que ofrece esta librería es la generación y manipulación de nubes de puntos. Las nubes de puntos son un conjunto de puntos en 3D (con tres coor-

denadas) que definen un cierto espacio. Como ya se mencionó, ésta era precisamente la funcionalidad que se buscaba, ya que se pretendía poder virtualizar el espacio interior con el dispositivo Kinect, y las nubes de puntos ofrecían la opción de obtener una "virtualización en 3D" que no consumía mucha memoria, y que a su vez permitían el análisis del espacio al conocer las coordenadas de los puntos.

La idea es sencilla, si se dispone de un conjunto de nubes de puntos en 3D guardado en memoria. Para establecer posteriormente la posición dentro de ese espacio a través de una imagen, habría que encajar dicha imagen dentro de la nube de puntos y con eso obtendríamos la posición de la persona. Para esto se estudiaron las diferentes posibilidades que ofrece PCL en el tratamiento de nubes de puntos:

- **Captura de nubes de puntos:** PCL consigue nubes de puntos a través del dispositivo Kinect gracias al uso de la librería auxiliar OpenNI que se instala con PCL. Se estudió la calidad de las nubes de puntos que se obtenían, si definían bien el espacio y el resultado fue satisfactorio. Las nubes de puntos se obtenían al instante, obteniendo una frecuencia aproximada de 60Hz en el refresco de captura, y la calidad de las mismas era aparentemente aceptable. En la figura 37 se pueden observar algunas de las nubes de puntos obtenidas con esta librería. La primera imaegn a color corresponde con una fotografía real de la estantería. El resto de imágenes en azul y negro representan capturas de las nubes de punntos vistas desde diferentes ángulos.

Figura 23: Nube de puntos obtenida con Kinect

- **Filtrado de nubes de puntos:** En muchos casos las nubes de puntos obtenidas contienen puntos repetidos, o puntos que no aportan información adicional y consumen memoria innecesariamente. Estos puntos pueden ser eliminados y PCL dispone de diversos filtros que los eliminan. Este tipo de funcionalidad es muy im-

portante, ya que si se quiere guardar una nube de puntos del espacio virtualizado, cuanto menor sea su tamaño, más efectiva será después la búsqueda. Dicho de otro modo, habrá que comparar la imagen que se quiere localizar con un menor número de puntos, lo que aceleraría la comparación. Además, como estos puntos eliminados no aportan nueva información, no se están perdiendo opciones de encontrar el encaje deseado. Estos son algunos de los filtrados que se aplicaron en las pruebas:

- **Filtro PassThrough**: Con este filtro lo que se consigue es desechar los puntos que se encuentran más allá de una profundidad que el propio usuario puede definir. De esta manera, en un ejemplo real en el que tenemos dos personas en nuestro encuadre, una de ellas a 2 metros de la pared y otra pegada a la pared, utilizando este filtro nos quedaríamos sólo con los puntos que componen a la persona que se encuentra más cerca del aparato.

  Consideramos que podría ser útil para nuestro objetivo porque también se puede utilizar en sentido contrario y descartar los puntos más cercanos a la cámara del Kinect ajustando el intervalo correspondiente. Así, nos quedaríamos tan sólo con la pared y podríamos descartar elementos que se interpusieran en la escena y que más adelante podrían complicar el tratamiento de la nube de puntos.

- **Filtro VoxelGrid**: Este otro filtro se basa en un procesamiento especial mediante voxels para disminuir la cantidad de puntos que componen la nube. Un voxel es la unidad mínima procesable en un objeto 3D, y es por tanto equivalnte al píxel. Este filtro traza por tanto una malla de voxels de manera que identifica los puntos de la nube con los voxels y es mas sencillo reconocer formas y objetos. De esta manera consigue conservar la densidad de puntos en las zonas de importancia, como bordes y contornos de objetos y prescindir de datos redundantes como superficies lisas. Como se explica anteriormente, para los cálculos que se realizan más tarde no es necesario tener una gran densidad de puntos en la nube. Ajustando algunos parámetros, este filtro es capaz de disminuir la densidad todo lo que el usuario necesite.

  Es realmente importante este filtro ya que consigue que los archivos que guardan las nubes de puntos ocupen mucho menos espacio y sean más manejables. Se puede ver un ejemplo de lo eficiente que es el filtro en la figura 24. La mesa de la izquierda es la nube de puntos antes de filtrar y la de la derecha la nube de puntos filtrada. Podemos obsevar que la aunque densidad de puntos ha disminuido considerablemente, no se ha perdido la estructura de los objetos de la escena.

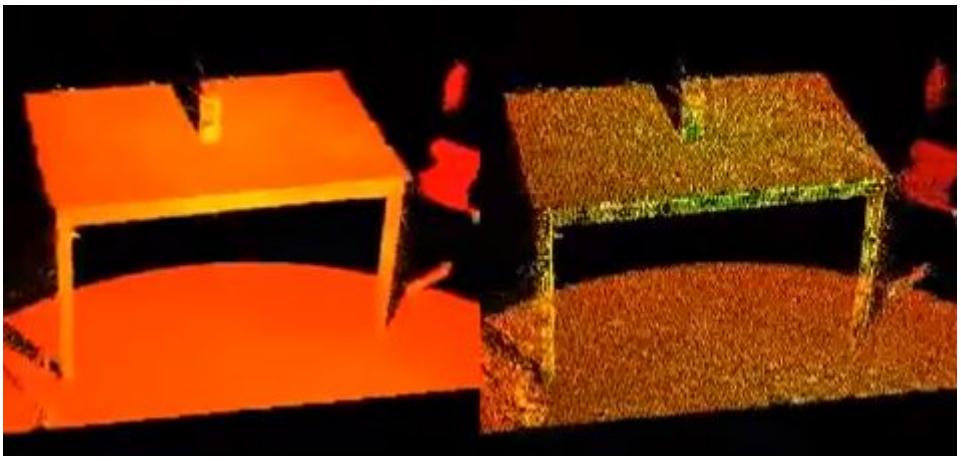

Figura 24: Filtro VoxelGrid

- **Filtro StatisticOutlierRemoval**: Con este último filtro se eliminan los puntos denominados outliers. Son aquellos que no forman parte de ningún plano de la escena y por ello no tienen ninguna utilidad para el procesamiento de la nube de puntos. Al eliminar estos puntos se consigue que la nube ocupe algo menos de espacio en disco, y que a la hora de procesarla no se invierta tiempo en ellos.

- **Point cloud registration:** Esta técnica consiste en la fusión de varias nubes de puntos en una sola. Era un punto clave para el desarrollo del proyecto, puesto que la idea final era la de disponer de una sola nube de puntos (o de un conjunto limitado de ellas) que describieran el espacio virtualizado. Dado que el dispositivo Kinect obtiene nubes de puntos dentro de su FOV era necesario ir uniendo cada nube de puntos con la siguiente, haciendo una fusión de puntos entre ambas nubes, para ir avanzando por el espacio y conseguir una única y coherente nube de puntos. Para este cometido PCL utiliza funciones basadas en el algoritmo ICP que busca continuamente puntos "muy cercanos" en las dos últimas capturas que pueda considerar como el mismo, y a partir de ahí unir ambas nubes de puntos. Se realizaron varias pruebas con las herramientas que ofrece PCL para esta tarea, pero no tuvieron éxito. Estas herramientas funcionaban bien si las nubes de puntos que se fusionaban tenían un conjunto de puntos en común en torno al 15 % de la imagen. Cuando se fusionan dos nubes de puntos se duplican algunos puntos en la nube final puesto que no se han encontrado coincidencias. Cuantos más puntos en común tengan las dos nubes, más puntos se duplican y la nube resultado comienza a perder la forma puesto que se vuelve muy densa y con gran cantidad de puntos erróneos. En nuestras reconstrucciones la captura de nubes se hace a una velocidad elevada, y habría que controlar muy bien cuándo se fusiona una nube de puntos con la siguiente para que este fenómeno no se produzca.

Kinfu

Kinfu es la herramienta que ofrece PCL inspirada en Kinect Fusion, una aplicación de Microsoft Research, para la virtualización de un espacio directamente en una malla poligonal 3D. A partir de las nubes de puntos "estima" posibles líneas que definen los

polígonos tridimensionales. Es una funcionalidad que se encuentra ahora mismo en desarrollo y que no está ni mucho menos terminada. Se probó su funcionamiento y el problema que tenía es que perdía fácilmente las conexiones o referencias entre una imagen y la siguiente, terminando por no funcionar correctamente en espacios relativamente grandes, como un piso por ejemplo. No obstante ofrece buenos resultados en ciertos espacios mas pequeños como una habitación, y es muy posible que en un futuro pueda llegar a funcionar bien y ser una buena alternativa.

De todas maneras, Kinfu no ofrecía la funcionalidad específica que se requería en este proyecto puesto que lo que se necesitaba eran nubes de puntos, o algún otro sistema que permitiera la posterior comparación para situar una imagen dentro del espacio (la localización en el interior) y los modelos 3D que obtiene Kinfu no pueden ser utilizados para este fin. Aunque sí que podría ofrecer una buena solución a la hora de visualizar el espacio en 3D sin tener en cuenta la localización, y podría ser útil para este fin posteriormente.

Conclusiones

PCL es una librería muy potente para todo tipo de tratamiento de nubes de puntos, nos permitía obtenerlas a partir del sensor Kinect, y luego manipularlas. Sin embargo no resultaba sencillo el hecho de componer todas las nubes de puntos obtenidas en una sola, y sin dicha funcionalidad, no era muy útil.

Tampoco ofrecía ningún mecanismo para la localización, que es el mayor reto de este proyecto. PCL no ofrece la funcionalidad necesaria para dada una imagen y una nube de puntos, conseguir situar dicha imagen dentro de la nube de puntos. Por esto también descartamos el uso de esta librería en nuestro proyecto, ya que no podíamos conseguir una buena virtualización del espacio al no poder componer nubes de puntos y tampoco localizarnos dentro de ellas, los dos objetivos primarios del proyecto.

3.2. ACG_LOCALIZER

Después de comprobar que PCL no aportaba técnicas de la localización, es decir, nos permitía generar nubes de puntos pero en ningún caso nos ofrecía funcionalidad para poder comprobar si una imagen pertenecía o no a una nube de puntos, decidimos investigar proyectos específicamente orientados a ese objetivo. Es así como encontramos el proyecto de Torsten Sattler, Bastian Leibe y Leif Kobbelt de la Aachen University [16]. Este artículo habla de métodos para calcular la posición de una imagen dentro de una nube de puntos utilizando diferentes técnicas. Parece que es exactamente lo que necesitaba el proyecto, pero se encontraron ciertos inconvenientes que se explicarán más adelante.

Introducción

Esta investigación presenta una solución a la localización basada en imágenes. A partir de un gran conjunto de fotografías (dataset) que describen un espacio, se consigue una

reconstrucción en 3D de dicho espacio. A este preproceso se le denomina Bundler. Este artículo se apoya en otros proyectos que proporcionan el preprocesado Bundler a partir del dataset. Con la reconstrucción extraída del dataset se procede al siguiente paso que consiste en generar visual words (la investigación fundamental del artículo). Estas visual words son identificadores que nos permiten clasificar los puntos más parecidos para realizar búsquedas más eficientes. A continuación en la subsección "Cálculo de asignaciones de descriptores" se explica más detalladamente este proceso.

La idea de uso de este sistema es más o menos sencilla. El gran reto de la localización basada en imágenes es que se requiere, como ya se ha comentado, grandes datasets de imágenes con los que realizar la reconstrucción 3D y a partir de ahí buscar en dicho dataset las partes de la reconstrucción (conjunto más pequeño de imágenes) que sean mas similares a la imagen que se quiere localizar. La búsqueda sobre el dataset de la reconstrucción puede ser muy grande, y esto minimiza muchísimo el rendimiento, haciendo incluso inviable la implementación de estos sistemas. La solución que se plantea aquí es el uso de los visual words mencionados antes, cuya misión es agilizar la búsqueda en estos enormes datasets.

Descripción del proyecto

En esta sección se explicará un poco más a fondo cada una de las fases de este proyecto, y en qué medida se relaciona cada una con la fase de investigación de ANVM. Se adjunta también en la figura 25 un esquema resumen del proyecto Acg localizer con el que se pretende facilitar la comprensión de lo que a continuación se expone. Este esquema está disponible en la página web del proyecto "http://www.graphics.rwth-aachen.de".

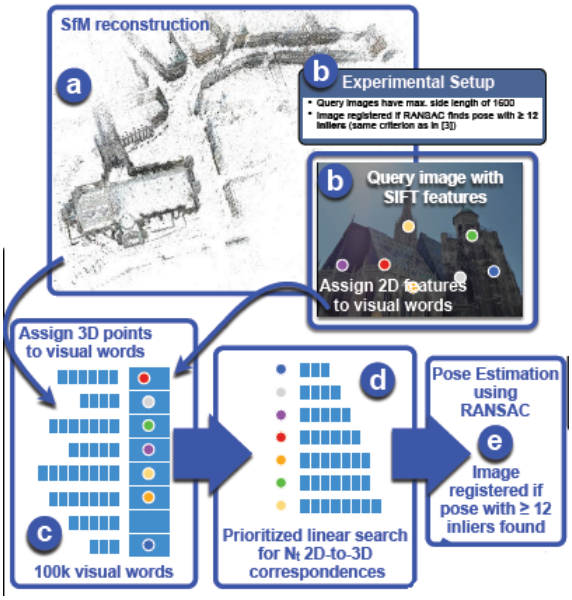

Figura 25: Esquema resumen del proyecto Acg localizer

Bundler

Como ya se ha explicado anteriormente en la localización basada en imágenes. Este proceso se corresponde con la obtención de una reconstrucción tridimensional a partir de un conjunto de imágenes. Para este cometido el Acg localizer utiliza otro proyecto de investigación que implementa estos algoritmos, en concreto, Bundler SfM de Noah Snaverly descrito anteriormente. Al final de este proceso se obtiene un fichero de salida con la información de la reconstrucción a partir del conjunto de imágenes de entrada, que (formateado de la forma correcta) se utiliza en la siguiente fase.

La idea inicial en nuestro caso era utilizar el dispositivo kinect para la obtención de dicha reconstrucción 3D, posteriormente darle el formato necesario a la salida y así continuar con el proceso que describe este proyecto. No obstante más adelante se observó que esto no era posible, tal y como se explica en la siguiente sección.

Cálculo de asignaciones de descriptores

Ésta es la fase clave del proyecto Acg localizer. En este punto es donde se introduce la mejora mencionada anteriormente: las Visual Words (vw). A partir del archivo generado por el bundler, se realiza una asignación de los descriptores de las imágenes (puntos 3D) a las llamadas visual words. Estas vw no son más que identificadores que nos sirven para clasificar los descriptores y de esta manera realizar búsquedas más eficientess sobre dichos descriptores. Las vw son obtenidas mediante un proceso de agrupamiento (clustering) de los descriptores de la imagen que se desea localizar dentro de la reconstrucción. Se agrupa dentro de una misma vw un conjunto de descriptores (de la imagen de entrada) que tienen un cierto parecido entre sí (la distancia euclídea entre sus componentes sea la menor posible), de manera que después la asignación de dichos descriptores a los puntos 3D de la reconstrucción sea mucho más eficiente. Recordamos que un descriptor SIFT es un vector de 128 números en coma flotante, aplicando el método de k-nearest neighbors según la distancia euclídea entre todos los vectores de descriptores, de esta manera generamos grupos o *clusters* de k descriptores.

Cada grupo o *cluster* viene representado por un solo descriptor correspondiente al "centro del grupo" (vw) calculado mediante el método kmeans. Este método consiste en dividir un cierto conjunto de elementos en clases con la menor distancia entre ellos posible y se trata de un problema NP-difícil, es decir que no se conoce un algoritmo que consiga resolver este problea en tiempo polinómico. Sin embargo existen algoritmos heurísticos bastante eficientes que permiten calcular estos centros del grupo. Se investigaron diferentes librerías para realizar kmeans dado que el cálculo sobre varios millones de descriptores puede llegar a tardar días si no se dispone de métodos eficientes. Entre las mejores librerías estudiadas encontramos mc-kmeans que implementa el algoritmo explotando su carácter paralelo ejecutandolo en varios threads. Por otro lado, approximate kmeans es una variante que consigue resultados casi óptimos en un tiempo de ejecución menor que el anterior.

Una vez que se dispone de los cluster centers (vw) de la imagen de entrada se procede a calcular la asignación de cada punto 3D de la reconstrucción a una vw, aquella a la que se parezca más, es decir la distancia euclídea entre el descriptor del punto 3D y el

cluster center sea la menor. Se genera un archivo con estas asignaciones y se procede a la siguiente fase.

El problema en este punto es que en realidad, para asignar los puntos 3D de la reconstrucción a las visual words, se utilizan los descriptores de los puntos 2D del dataset de imágenes de entrada. Es decir, que se utilizan los descriptores de las imágenes de la reconstrucción, para asignar el punto que le corresponde en 3D a cada visual word. Dado que nuestra idea era utilizar, en vez de un dataset de imágenes de entrada, una reconstrucción de nubes de puntos 3D ya calculada previamente con el Kinect no disponíamos de los puntos 2D de esas imágenes de entrada. Esto no nos permitía realizar esa asignación entre puntos 2D - 3D que aquí se utiliza.

Localización

Una vez que se dispone de las asignaciones entre los puntos 3D de la reconstrucción y los 2D de la imagen que se desea localizar, calculado gracias a las vw en el apartado anterior, tenemos todos los datos necesarios para calcular la posición exacta del usuario que ha realizado la foto (imagen de entrada). Esta posición se calcula utilizado un algoritmo basado en RANSAC (RANdom SAmple Consensus). Este procedimiento consiste en ir probando diferentes posiciones aproximadas, comprobando lo bien que se ajustan a las correspondencias 3D - 2D. Para cada muestra se calcula el número de correspondencias que se cumplen, denominadas *inliers*.

En el artículo de este proyecto se adjuntan varios experimentos realizados, y los resultados obtenidos son bastante buenos, el error de la posición calculada es aceptable (error traslación de unos pocos centímetros y el error angular menor que 5º). También se detallan aspectos sobre cómo configurar cada uno de los parámetros del proyecto (número de inliers para RANSAC, número de árboles a utilizar en la búsqueda de correspondencias...) así como aquellos valores con los que se obtienen los mejores resultados.

Conclusiones

Este proyecto resuelve de una manera bastante eficiente el problema de la localización basada en imágenes. Es capaz de averiguar la posición y dirección del observador que ha tomado la imagen dentro de la reconstrucción con un error muy aceptable para nuestros propósitos. Sin embargo, y a pesar de que propone una implementación más o menos eficiente, este proyecto es difícilmente aplicable a dispositivos móviles. Todo el proceso de reconstrucción no cuenta con ese problema del rendimiento, porque ha de realizarse tan sólo una vez y puede realizarse en una máquina permitiendo tiempos más largos de ejecución. Sin embargo el cálculo de la posición y el de correspondencias 2D - 3D, a pesar de ser eficiente, es difícilmente implementable en un smartphone actual.

También se contemplaron opciones de implementar una arquitectura cliente-servidor para que los cálculos más pesados se ejecutaran en el servidor. No obstante las ideas investigadas de Acg localizer no han sido utilizadas en nuestro proyecto, aparte de por cuestiones de rendimiento ya mencionadas, por tratarse de un proceso realmente complejo que llevaría más tiempo en implementar del que se dispone para el proyecto. Por todo esto se utilizan otras técnicas de localización diferentes de la localiazación basada

en imágenes. Como se explica en el documento de memoria de ANVM: Mobile app [9], se han aplicado métodos de localización basados en otras técnicas como códigos QR.

Sin embargo no se descarta que en un futuro se pudiera ampliar el proyecto ANVM para que disponga de un sistema de localización basada en imágenes, similar al proyecto Acg localizer, lo que permitiría mayor autonomía a la hora de situarse en el espacio interior reconstruido.

3.3. ROS: Robot Operating System

Se trata de un gran conjunto de librerías para GNU/Linux destinadas a servir como base del desarrollo de todo tipo de aplicaciones relacionadas con la robótica [17]. Proporcionan gran funcionalidad en el manejo de dispositivos hardware permitiendo la abstracción por parte del programador a la hora de comunicarse con los dispositivos, así como multitud de de métodos y técnicas de gran utilidad en este campo. ROS es un proyecto de software libre regido bajo la licencia BSD, desarrollado por multitud de investigadores y científicos del campo de la robótica. Actualmente se continúan desarrollando nuevos paquetes para esta librería, tratando de añadir cada vez más funcionalidad según se va avanzando también en la investigación en este campo en continuo crecimiento.

Relación de ROS con el proyecto ANVM

En este punto de la investigación, se había dejado a un lado la localización basada en imágenes, tal y como se explica en los apartados anteriores. Por tanto, en esta librería se busca un objetivo diferente, la reconstrucción de un espacio interior, pero ya no interesa su reconstrucción 3D, sino la obtención de un mapa 2D que sirva de "plano" para la aplicación móvil desarrollada en el proyecto paralelo ANVM: Mobile app.

Se buscan por tanto en esta librería herramientas que faciliten obtener, a partir de las nubes de puntos adquiridas con el dispositivo kinect, datos parecidos al de un escáner láser en 2D (sistema LIDAR ver glosario). La idea consiste básicamente en que si tenemos una reconstrucción en nubes de puntos 3D del espacio, con realizar un corte horizontal paralelo al suelo obtendríamos un plano 2D similar al de la figura 26 en el que se pueden distinguir las paredes del espacio interior disponible. El área en gris claro sería el correspondiente al suelo, o espacio interior, y estaría delimitado por las paredes que aparecen en negro. La posición actual del escáner se distingue por los ejes que aparecen en rojo y verde, y los puntos en rojo los que estaría actualmente capturando.

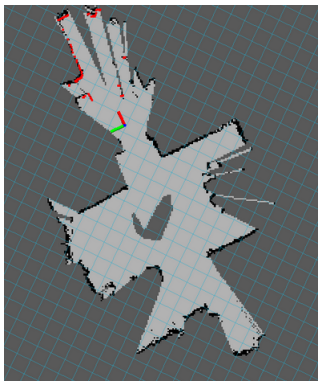

Figura 26: Plano 2D generado por un escáner láser

Para llevar a cabo este nuevo planteamiento del problema de la reconstrucción, se estudiaron los paquetes de esta librería que tenían que ver con todo el tema de lo que se demonima SLAM descrito en la sección 1. A continuación se detallará cada uno de los aspectos estudiados de esta librería y cuales de ellos fueron útiles o no para el proyecto.

Utilización de la librería

Existen varias distribuciones de la librería ROS, para la investigación de este proyecto se probaron dos: ROS Fuerte y ROS Electric. La versión Fuerte es más actual que la Electric y se utilizó al principio de la investigación, aunque rápidamente se descartó debido a que no disponía de todos los paquetes que eran necesarios, bien porque se habían eliminado, o porque otras librerías en las que se apoya ROS no tenían todavía una versión preparada para esta distribución. La mayoría de las pruebas se realizaron por tanto en la distribución Electric.

ROS es una librería (o más bien, conjunto de librerías) muy extensa. Sin embargo es bastante sencilla de utilizar gracias a su división en paquetes. Cada paquete ofrece una cierta funcionalidad y se invocan mediante los ficheros .launch que se describirán a continuación. Para esta investigación se utilizaron los paquetes necesarios para la comunicación con un dispositivo que permitiera la captura de nubes de puntos (Kinect) y los necesarios para generar mapas 2D.

Ficheros “launch”

El uso de ROS se lleva a cabo mediante la ejecución de ficheros launch. Estos ficheros son documentos de texto escritos mediante un lenguaje de marcado que permite configurar cada uno de los paquetes de ROS que se quieren ejecutar. Se puede ver un ejemplo de este tipo de fichero en la figura 27. La etiqueta raíz del documento es <launch> y dentro de esta se van indicando cada uno de los componentes que se quieren lanzar, los identificados con la etiqueta <node>. Cada uno referencia al paquete que se desea ejecutar y los parámetros necesarios para la ejecución de dicho paquete.

```
<launch>
  <!-- kinect and frame ids -->
  <include file="$(find openni_camera)/launch/openni_node.launch"/>

  <!-- openni_manager -->
  <node pkg="nodelet" type="nodelet" name="openni_manager" output="screen" respawn="true" args="manager"/>

  <!-- throttling -->
  <node pkg="nodelet" type="nodelet" name="pointcloud_throttle" args="load pointcloud_to_laserscan/CloudThrottle openni_manager">
    <param name="max_rate" value="2"/>
    <remap from="cloud_in" to="/camera/depth/points"/>
    <remap from="cloud_out" to="cloud_throttled"/>
  </node>

  <!-- fake laser -->
  <node pkg="nodelet" type="nodelet" name="kinect_laser" args="load pointcloud_to_laserscan/CloudToScan openni_manager">
    <param name="output_frame_id" value="/openni_depth_frame"/>
    <remap from="cloud" to="cloud_throttled"/>
  </node>
</launch>
```

Figura 27: Fichero .launch

rviz

Es la herramienta de la que dispone ROS para visualizar la ejecución. Muestra el resultado visual de los ficheros launch que se encuentran en ejecución. Es una herramienta muy útil que permite la configuración de todo lo que se quiere visualizar. Durante las pruebas de la librería se utilizó esta herramienta y fue de gran utilidad para probar su funcionamiento. Se puede ver una captura de esta herramienta en la figura28. En la sección de la izquierda se nos permite configurar los parámetros referentes a la captura que se está realizando. En el centro dela pantalla vemos la captura de la reconstrucción actual.

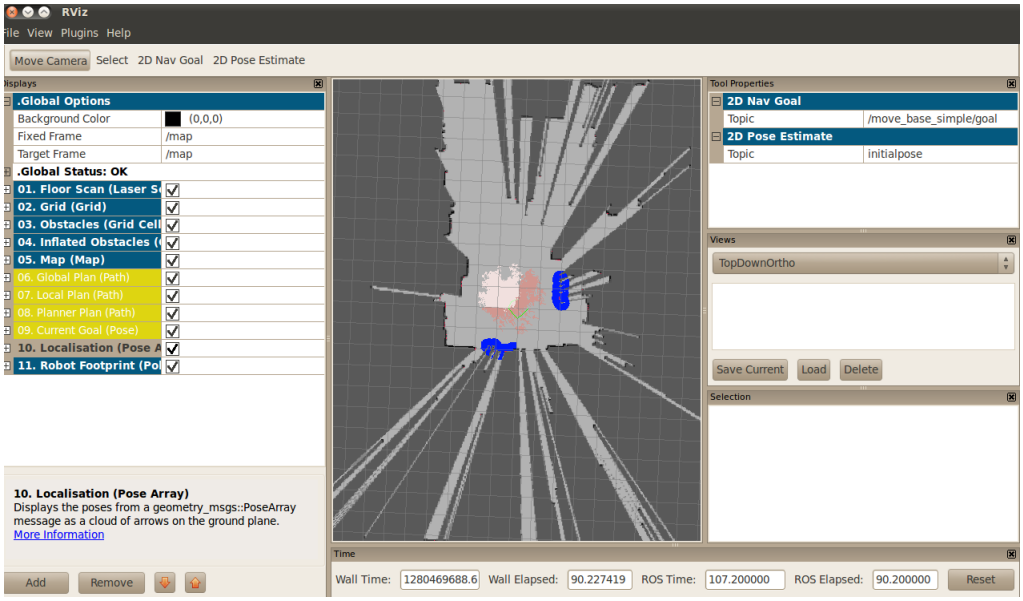

Figura 28: 2D slam con la herramienta rviz

2D SLAM con ROS y Kinect

El Dr. Rainer Hessmer ofrece en su blog un post sobre como utilizar la librería ROS y Kinect para realizar una reconstrucción de un espacio interior y obtener un plano del lugar reconstruido (2D SLAM). Mediante los ficheros launch descritos anteriormente se simula un escáner láser 2D LIDAR por medio del dispositivo Kinect. Esto nos ofrece un

conjunto de puntos con sus diferentes profundidades que hay que ir uniendo mientras se va renconstruyendo el espacio. Para unir dichos puntos y conseguir formar el mapa que se muestra en la figura 26 es necesario tener información sobre la posición del Kinect en cada momento para saber cómo unir dichos puntos. Es en este momento de investigación cuando los desarrolladores de este proyecto nos dimos cuenta de que necesitamos otro dispositivo a parte del Kinect que nos suministre esta información.

Para este proyecto se realizaron pruebas basándose en el post del Dr. Hessmer. En primer lugar se ejecuta el fichero .launch (figura 27)que se encarga de recoger los datos del kinect. Las pruebas en este sentido fueron satisfactorias, sin embargo en el siguiente paso el resultado no fue tan favorable. Como se indica en la sección 1.2 para realizar un 2D SLAM al aplicar las técnicas de composición de nubes de puntos no se obtienen resultados igual de buenos que para 3D SLAM y es necesario la utilización de información adicional que nos indique desde qué lugar exacto de la reconstrucción se ha realizado la captura de cada nube de puntos. El Dr. Hessmer propone, en su post, la utilización de un robot fabricado por él mismo que sea capaz de suministrar la información de odometría (posición) necesaria para realizar la reconstrucción.

Este robot dispone de un sistema capaz de calcular la posición actual del robot a partir de un punto de partida, y utiliza como procesador un microcontrolador ArduinoUno del que hablaremos más adelante (ver sección 3.4). En este punto de la investigación, ANVM no contaba con dicho robot capaz de dar información de odometría, por lo que no se pudo probar el funcionamiento de este 2D SLAM propuesto por el Dr. Hessmer. A partir de aquí, la investigación se centró diseñar un robot similar al que se propone aquí, para captar la información sobre odometría.

Conclusiones

En definitiva ROS es una herramienta que ofrece tremenda funcionalidad. Para este proyecto podría tratarse de una librería muy útil, ya que nos facilita la funcionalidad que requiere como herramientas para 2D SLAM. Sin embargo se trata de una librería extremadamente potente y nada sencilla de comenzar a utilizar. Posee una gran cantidad de módulos y un sistema de configuración de los mismos mediante los fichero .launch que son del todo desconocidos por los integrantes del equipo. Requeriría una gran inversión en tiempo de formación para comprender el funcionamiento de esta librería, tiempo del que no se dispone para este proyecto. Debido a esto se ha decidido no utilizar esta librería. No obstante se podría en un futuro contemplar utilizar ROS como complemento del proyecto dado que dispone de elementos interesantes para futuras amplicaciones.

3.4. Arduino

Arduino es una plataforma open hardware de prototipado electrónico creado en 2005 por Massimo Banzi y David Cuartielles. Ofrece un entorno de desarrollo completo y además varias alternativas hardware también open source. Sin embargo su característica principal reside en la gran accesibilidad que proporciona a usuarios no experimentados o sin grandes conocimientos de electrónica. Gracias a ello, su popularidad en los últimos años ha ido creciendo a un ritmo sorprendente, utilizado por entusiastas y curiosos en

pequeños proyectos, hasta incluso como herramienta de enseñanza en universidades. A su paso ha ido dejando una comunidad muy colaborativa y una multitud de librerías y programas ejemplo de código libre.

Al reconocer el requisito de diseñar nuestro propio sensor, para así poder detectar la posición del Kinect al realizar las capturas, la primera idea fue emplear hardware ya conocido por su uso en la carrera, como por ejemplo una FPGA o un microprocesador ARM. Esta idea inicial fue rápidamente descartada una vez investigamos otras opciones y descubrimos Arduino. Su facilidad de uso y extensa documentación la pusieron por delante de las otras opciones (FPGA y ARM) mucho más complejas y costosas. A continuación se explican brevemente las características de la placa Arduino Uno, empleada en el proyecto por su bajo coste, amplio repertorio de conexiones y funcionalidad, además de su reducido tamaño.

Arduino Uno

La placa Arduino Uno (ver figura 29) es la más común del mercado, con un precio de aproximadamente 30€ y unas dimensiones de 6.85 x 5.33 cm. es ideal para pequeños proyectos de robótica y control automático [2]. Basada en el microcontrolador ATmega328, operando a una frecuencia de reloj de 16MHz y 32KB de memoria flash ofrece una gran versatilidad para desarrollar proyectos gracias a un total de 20 pines de entrada/salida, 16 digitales y 4 analógicos. Cuenta también con una memoria EEPROM de 1KB y otra memoria SRAM de 2KB. Requiere una alimentación de 5 voltios, ideal para conexión via USB o batería externa. Por otro lado dispone de un chip dedicado para transmisión de datos mediante USB emulando una conexión puerto serie. Esto facilita enormemente la comunicación hacia y desde el dispositivo.

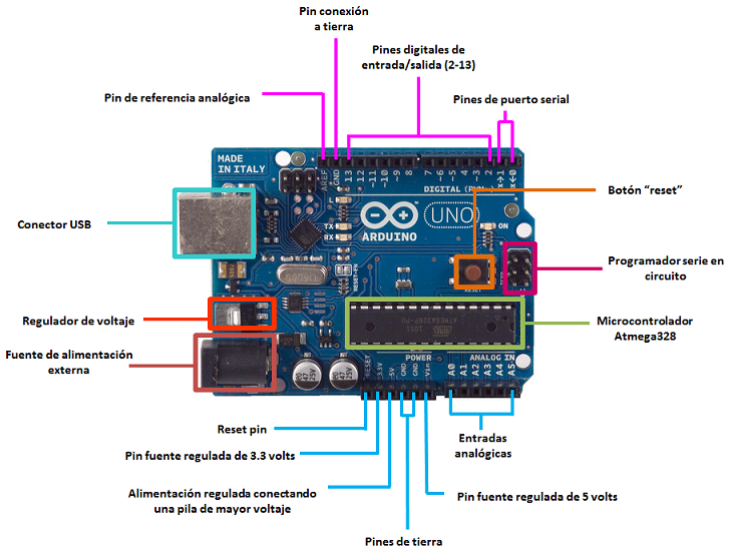

Figura 29: Arduino Uno

Finalmente cabe destacar la abundante selección de placas de expansión (llamadas Shields) que podemos encontrar en el mercado: conexión Wifi, LAN, acelerómetros,

giroscopios, ultrasonido, displays LCD, etc. El uso de estas expansiones convierte al Arduino Uno en una plataforma de desarrollo extremadamente pontente y versátil.

Programación en Arduino

Arduino cuenta con un entorno de desarrollo integrado que permite la escritura, compilación y carga de programas arduino llamados sketches. Construido sobre java, tiene la ventaja de ser multiplataforma y fácilmente ampliado al ser código open source. En la figura 30 podemos ver una captura del entorno de desarrollo.

La sintaxis de arduino es muy simple, siguiendo un paradigma de programacion imperativa, consta de los típicos operadores aritméticos, declaración de funciones y estructuras de control (if-then-else, bucles, ...). Además incluye varias instrucciones específicas de entrada/salida digital y analógica. Por otro lado existen numerosas librerías que ofrecen funcionalidad para conectividad Wifi, lectura de tarjetas SD, control de servos, matrices de LEDs, etc.

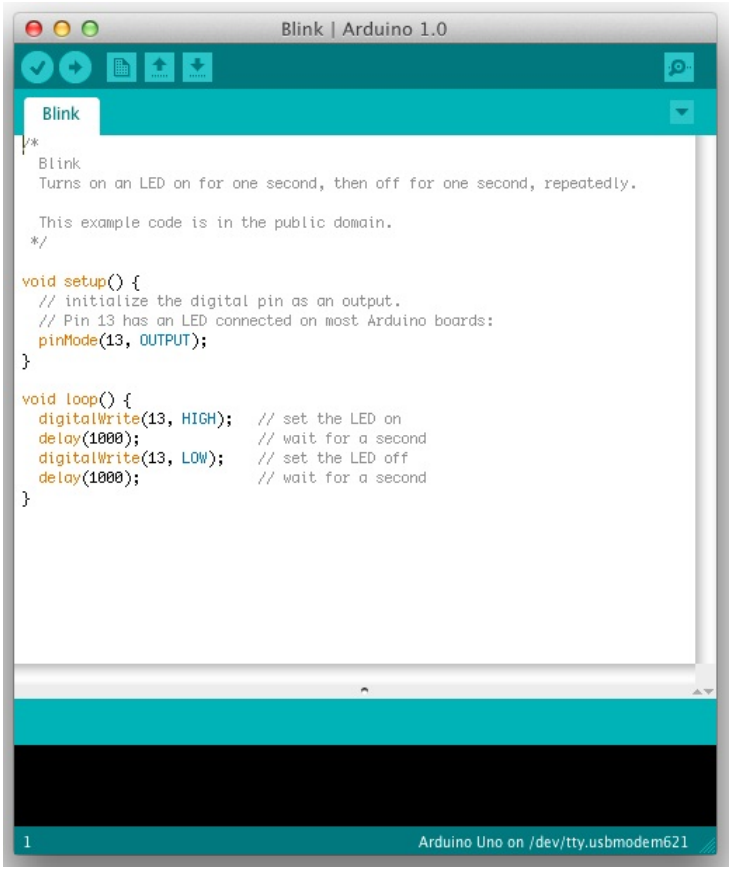

Figura 30: Entorno desarrollo integrado para arduino

3.5. MRPT: Mobile Robot Programming Toolkit

Se trata de una librería escrita en C++ desarrollada por investigadores de la universidad de Málaga que proporciona una extensa API de funciones de gran utilidad para

el campo de la robótica [18]. Su función es similar a la librería ROS comentada anteriormente, proporcionar un entorno con un conjunto de funcionalidad para el desarollo de aplicaciones relacionadas con la robótica. Ofrece algoritmos ya implementados para problemas de localización, SLAM, y evitación de obstáculos, muy útiles para el diseño de robots.

Esta librería funciona tanto el GNU/Linux como en Windows ya que una de sus características reseñables es su portabilidad y reusabilidad. Es una librería de código libre regida bajo licencia BSD y cuenta además con una extensa documentación, algo muy bienvenido para desarrolladores noveles en el campo de la robótica. Propone una serie de tutoriales para comenzar a utilizar la librería, así como una serie de aplicaciones ya implementadas de gran utilidad para este proyecto. Se apoya, como el resto de librerías, en otra librería (OpenNI) para el manejo de la entrada y salida de datos provenientes de dispositivos hardware como el Kinect.

MRPT permite el desarrollo de aplicaciones en un entorno C++, que es un lenguaje bien conocido por los desarrolladores de este proyecto. Esto fue algo que se tuvo en cuenta a la hora de elegir esta librería como la definitiva a utilizar para el desarrollo del proyecto. A continuación se van a comentar cada uno de los aspectos investigados de esta librería.

Ficheros RAWLOG

Es el principal tipo de fichero que maneja esta librería. En estos ficheros se almacena toda la información que se puede generar, y se ofrecen funciones tanto para leer de ellos como para escribirlos. Estos ficheros son cajones de sastre donde se puede guardar cualquier objeto de la librería MRPT, desde capturas de escenas 3D hasta información de odometría del robot. Durante la realización de las pruebas sobre la librería se utilizaron continuamente este tipo de ficheros para guardar las capturas de datos que se tomaban con el Kinect.

MRPT dispone también de un conjunto de herramientas útiles para el desarrollador como, por ejemplo, el RAWLOG viewer (ver figura 31). Mediante esta herramienta se puede observar el contenido de un rawlog generado al realizar una reconstrucción (lista de observaciones a la izquierda de la imagen), y nos permite inspeccionar cada una de las tomas que se han realizado (el resultado de la toma son los puntos azules que se observan abajo a la derecha de la imagen, en este caso un láser 2D). Tanto las nubes de puntos, como las imágenes (vídeo) y las coordenadas de la odometría que se estén utilizando en cada paso. También permite animar la reconstrucción realizada para ver el resultado obtenido mediante las opciones del menú superior.

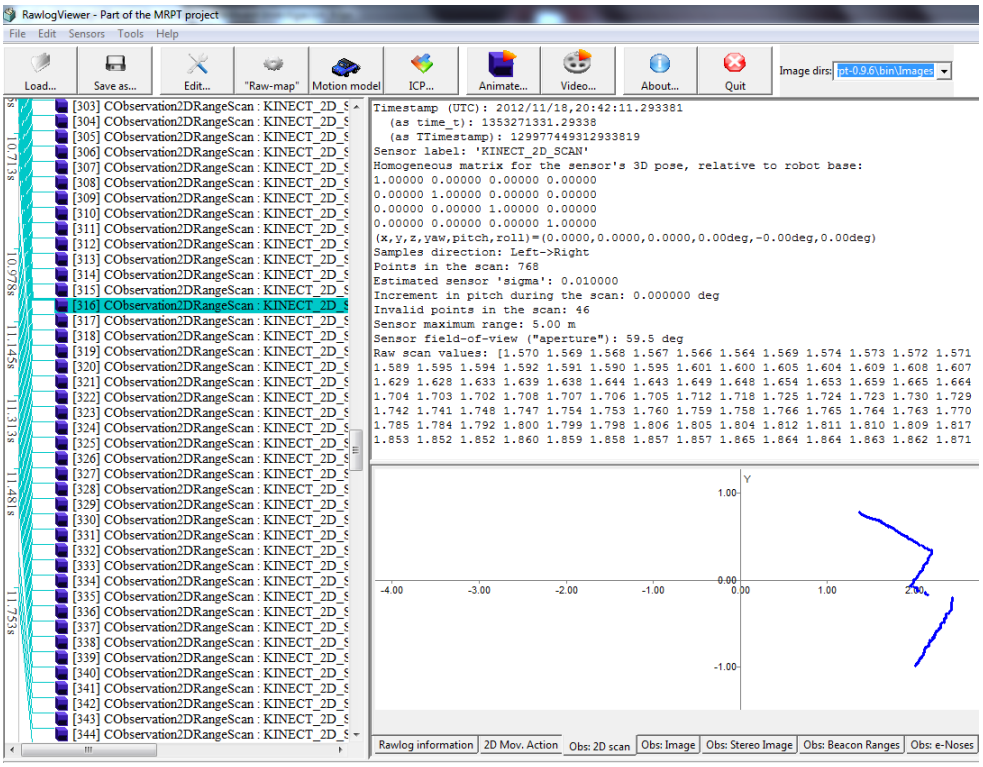

Figura 31: RAWLOG viewer: Visión de un fichero RAWLOG

Depende de lo que se desee guardar en este tipo de ficheros el tamaño del rawlog puede variar considerablemente. En las pruebas realizadas, para la virtualización de una habitación los rawlog varían desde los 160 MB hasta los 800 MB dependiendo de si se está realizando 2D SLAM o 3D SLAM respectivamente. Cabe imaginar que el tamaño es algo grande como para plantearse hacer una reconstrucción a gran escala. No obstante durante las pruebas se observó que en el RAWLOG se puede decidir exactamente qué elementos de la reconstrucción se desean guardar y se consiguieron ficheros de tamaño menor de 1 MB para una habitación. Esto es debido a que en un primer momento se guardaba absolutamente toda la información capturada por el Kinect. Esto incluye un vídeo de la captura, todas las imágenes por separado, información de nubes de puntos 3D, etc. Posteriormente se eliminaron datos innecesarios para la reconstrucción como son el vídeo y las imágenes y dejando sólo las observaciones transformadas a 2D que son las que se utilizan en la reconstrucción, reduciendo el tamaño del fichero considerablemente.

Ficheros INI

Consisten en ficheros de configuración o inicialización que son empleados por las aplicaciones de ejemplo de la librería. Sirven para ajustar cada uno de los parámetros de la aplicación que se va a lanzar. Estos ficheros difieren dependiendo de cada aplicación pudiéndose consultar las plantillas a seguir para cada una en la página web de MRPT. Desde estos ficheros es posible ajustar un gran rango de configuraciones, desde los parámetros individuales que utiliza cada algoritmo hasta la distancia estimada entre cada toma realizada para el SLAM.

Aplicaciones de ejemplo

Para comprobar que esta librería se ajustaba a las necesidades del proyecto, se investigaron algunos de los ejemplos (aplicaciones) que propone MRPT y que se detallarán a continuación.

Kinect 3D SLAM

Se trata de una aplicación bastante sencilla con muy pocas líneas de código que realiza una virtualización de un espacio. Se trata tan sólo de un ejemplo y por ello no debe sorprender su mal funcionamiento. No está recomendado su uso para mapas ni siquiera de un tamaño medio debido a su escasa robustez en la reconstrucción. Pierde la posición estimada con bastante facilidad.

Las pruebas realizadas con este ejemplo muestran que difícilmente se puede obtener una reconstrucción de una simple habitación, dado que entre una toma y otra muchas veces la aplicación confunde la posición actual. Este problema ocurre durante la fusión o *matching* de las dos nubes de puntos sobre las que se está trabajando, entre la captura anterior y la actual se ha de encontrar un cierto número de inliers, o correspondencias entre ambas nubes. Esto no siempre se consigue y una vez que la aplicación se pierde, no tiene forma de volver a funcionar correctamente mas que reiniciándola.

A pesar del mal funcionamiento de este ejemplo, no hay que olvidar que no es más que eso, un ejemplo, que no cuenta con ningún tipo de corrección de errores u otro sistema para conseguir reubicarse una vez perdida. Este ejemplo demuestra la potencia de MRPT que tan sólo con unas pocas líneas de código bastante simples, es capaz de realizar un 3D SLAM (con mejor o peor resultado), un proceso nada trivial.

No obstante, dado que en esta etapa de investigación del proyecto interesaba más la obtención de un mapa 2D, no se le dio más importancia a este ejemplo, dejándolo como una posible referencia futura en caso de querer realizar un 3D SLAM.

RBPF SLAM

Durante la investigación de esta librería ésta fue la primera aplicación que se comprobó que realizaba un 2D SLAM. Las primeras pruebas no dieron buenos resultados ya que no se disponía todavía de información sobre odometría. El problema era que, al no conocer la posición desde la que se tomaba cada captura, no se podía reconstruir el espacio uniendo progresivamente las capturas de forma correcta, sino que erróneamente se amontonaban unas encima de otras.

Sin embargo en las siguientes pruebas que se realizaron se consiguieron ciertos avances. Dado que no se disponía todavía de un “robot” capaz de dar información sobre odometría, se intentó obtener esta información a partir de las imágenes. Se modificó el código de captura de ficheros RAWLOG (fichero que después se pasaría al rbpf slam) para obtener información sobre la odometría comparando una toma con la anterior, y deduciendo el movimiento que había realizado el Kinect. Los resultados obtenidos con

esta técnica fueron bastante mejores que los obtenidos anteriormente. Sin embargo todavía diferían bastante de la realidad. Se pueden ver ejemplos de este algoritmo en la sección de Plan de pruebas.

ICP SLAM

Esta aplicación genera un mapa 2D a partir de un fichero RAWLOG, es decir realiza un 2D SLAM utilizando el algoritmo ICP. El funcionamiento básico de este algoritmo (ICP) se ha explicado anteriormente y para cualquier duda se puede consultar el Glosario.

Tras el primer intento de reconstrucción con RBPF SLAM, dado que no fue satisfactorio por no disponer de información sobre odometría, se utilizó este método ya que en principio no la requiere para funcionar. Sin embargo el funcionamiento de esta herramienta sin odometría no dio buenos resultados al capturar superficies coplanares sin rasgos, ya que al no poder diferenciar una superficie coplanar de otra, los mapas se generaban superpuestos. Se pueden ver ejemplos de este algoritmo en la sección de Plan de pruebas.

Conclusiones

La investigación sobre esta librería fue satisfactoria ya que disponía de toda la funcionalidad que necesitaba el proyecto. Las pruebas realizadas permitieron ver que se podían conseguir los objetivos marcados para el proyecto y que por tanto MRPT era una buena elección como librería a utilizar. También se observó que el entorno de trabajo con esta librería era cómodo por varios motivos:

- En primer lugar, la existencia de una gran documentación sobre el código del proyecto ayuda en gran parte a la comprensión del mismo. Todas las clases y métodos están bien explicadas en dicha documentación así como consejos sobre su utilización. Al disponer también del código fuente de la librería fue más sencilla su utilización y entendimiento sobre qué está haciendo exactamente MRPT en los niveles inferiores de abstracción.
- Se trata de una librería desarrollada en C++. Esto permite una fácil integración en la aplicación ANVM desarrollada en el mismo lenguaje, muy conocido por el grupo del proyecto dado a experiencias previas durante toda la carrera con este lenguaje. Además, permite una buena integración con Arduino gracias a su sistema de control de threads, que simplifica mucho la toma de datos desde el sistema abordo del robot.
- Las aplicaciones de ejemplo ayudan en gran medida a comprender cómo utilizar la librería. Muchas de las tareas que se han de realizar para este proyecto están directamente relacionadas con una u otra aplicación de ejemplo. Esto permite poder consultar en todo momento cómo utilizar las clases de MRPT que intervienen directamente con el desarrollo del proyecto.

- Tratándose de una librería todavía en desarrolo y además de la universidad de Malaga, permite que exista comunicación con los desarrolladores de la librería para solucionar cualquier duda sobre el proyecto. Durante la investigación se consultaron dudas a través del foro y del correo electrónico siendo respondidas con bastante celeridad por parte de los autores de MRPT.

4. Conclusiones

El proceso de investigación de este proyecto ha sido largo y extenso, implicando la inversión de gran parte del tiempo disponible para el proyecto como se comenta en la sección de planificación. Sin embargo ha sido un proceso necesario ya que muchos de los conocimientos requeridos para realización de este proyecto no han sido adquiridos durante la carrera y ha conllevado su investigación previa a la implementación.

La investigación de ciertos campos ha requerido la modificación de algunos requisitos u objetivos que se habían puesto inicialmente al proyecto. Un ejemplo es el SLAM 3D, el cual se establecía como requisito en un primer momento hasta que se descartó después de investigarlo más a fondo. Otro ejemplo es la localización basada en imágenes, que tras realizar una exhaustiva investigación en esa dirección, se concluyó que no se disponía del tiempo necesario para su implementación.

Sin embargo no todo han sido conclusiones que merman los requisitos, también se han añadido nuevos, como por ejemplo la construcción de un robot Arduino. En un primer momento no se planteó la implementación de este sistema en el proyecto. Sin embargo tras realizar las averiguaciones pertinentes sobre el 2D SLAM, se concluyó que era un requisito viable y se añadió a los requisitos del proyecto.

Resumiendo el proceso de investigación, vamos a enumerar una serie de conclusiones a las que se llegaron cuando se finalizó dicho proceso y que han permitido el desarrollo del proyecto ANVM:

- **De 3D SLAM a 2D SLAM:** Se deja atrás la intención de reconstruir tridimensionalmente un espacio debido a su complejidad, y su funcionamiento no óptimo para una gran cantidad de superficies. Se opta en su lugar por tomar una mapa 2D (a escala) del espacio interior.
- **De Localización basada en imágenes a códigos QR:** Se abandona la intención de localizarse mediante una imagen (fotografía) tomada desde un punto cualquiera dentro de un espacio. Esto era un objetivo ambicioso y se contaba con no poder llevarlo a cabo. Las investigaciones en este campo dejaron ver que no se disponía del tiempo y los conocimientos necesarios como para realizar una implementación eficiente de esta técnica. En el proyecto paralelo ANVM: Mobile app se hablará más de la técnica que le sustituye empleando códigos QR.
- **Utilización del Kinect:** En un primer momento se pretendía su uso para captar nubes de puntos 3D para la realización del slam. Sin embargo, al optar por la alternativa del slam en 2D se plantea la utilización de este dispositivo como un emulador, más barato, de un escáner láser en 2D. Por tanto se sigue utilizando en este proyecto, pero con un enfoque diferente al que se pensó en un principio.

- **Arduino:** Este elemento del proyecto nació también del proceso de investigación. Según lo que se investigó en esta dirección, Arduino proporciona una interfaz sencilla para implementar el microcontrolador y simplifica mucho la obtención de la odometría.
- **MRPT:** Junto al sistema Arduino, esta es la librería sobre la que se soporta el proyecto ANVM. Se eligió después de investigar y experimentar exhaustivamente con otras tres librerías similares, prefiriendo su utilización por ser una herramienta potente y a la vez no muy complicada de utilizar. Ofrecía todo lo que el proyecto necesitaba y pudiera necesitar en un futuro.

ANVM es un proyecto, por tanto, al que le ha influido en gran medida la investigación ya que ha determinado en muchas cuestiones qué camino seguir y cómo adapatar la funcionalidad dependiendo de las averiguaciones que se hacían en cada campo. No obstante también ha permitido que los desarrolladores adquirieran un amplio conocimiento sobre el tema de SLAM, así como del resto de campos investigados, lo que facilitó en gran medida el posterior diseño e implementación de este proyecto.

Capítulo 4

Requisitos

Esta sección proporciona una vista general de la especificación de requisitos del proyecto ANVM. En ella se describen los requirimientos no funcionales, las restricciones de diseño y otros factores necesarios para proporcionar una completa y comprensiva descripción de los requisitos del software. Todas las palabras y siglas que necesiten una mayor explicación están recogidas en la sección Glosario.

1. Descripción general

En esta sección de los requisitos del sistema se describen los principales factores que afectan al producto, proporcionando un trasfondo para que en la siguientes secciones se definan con mayor claridad los requisitos.

1.1. Perspectiva del producto

La perpectiva del proyecto ANVM es la de conseguir llegar a todos aquellos usuarios que deseen obtener una aplicación móvil mediante la cual los clientes que puedan tener, ya se trate por ejemplo de un aeropuerto, una universidad o un centro comercial, sean capaces de sentirse cómodos, situados y con facilidad para llegar en todo momento a cualquier destino dentro del recinto.

Para ello se necesita disponer de un mapa de dicho recinto, y este es requisito principal de este proyecto. Se pretende construir el mapa para dar soporte a la aplicación móvil que disfrutará el usuario final.

Por otro lado se pretende que el proyecto pueda llegar a dar soporte para personas con discapacidad. Esto es, que la aplicación móvil sea capaz de comunicarse con el usuario que es ciego mediante la voz, guiándole e indicándole paso a paso el recorrido que debe hacer para llegar a su destino.

Además de la finalidad informativa de una orientación más comercial, ANVM sirve también para la creación de planes de evacuación de los lugares virtualizados. A petición del cliente que solicita el servicio, se puede desarrollar una extensión que consiste en la visualización de las rutas de evacuación en el mismo mapa junto con iconos que representan escaleras, salidas, extintores o mangueras de emergencia.

1.2. Funcionalidades del producto

- Virtualización de entornos interiores para la generación de mapas.
- Construcción de un robot capaz de llevar a cabo las virtualizaciones.

- Acondicionamiento del mapa generado para la aplicación móvil.
- Visualización del mapa de un complejo comercial o de servicios con varios niveles de zoom e iconos informativos.
- Búsqueda de puntos de interés en un complejo a través de una aplicación móvil.
- Localización dentro de dicho mapa mediante el escaneo de códigos QR.
- Obtención de rutas óptimas para desplazarse de un lugar a otro del complejo.
- Paseo virtual por el complejo mediante vistas panorámicas.

1.3. Características de los usuarios

Como se explica en la sección de Visión, dentro de los usuarios a los que está dirigida este sistema podemos distinguir dos tipos:

- **PYMES y empresas privadas o de servicios:** ANVM ofrece una posibilidad a empresas y PYMES de promocionar su negocio mediante la utilización de una aplicación que incluya los puntos de interés del complejo en cuestión así como un mapa por el que el usuario de la aplicación móvil pueda ser guiado. Estas empreseas podrán utilizar este producto para la generación de mapas de sus instalaciones.
- **Usuario final de la aplicación:** es aquel que utiliza la aplicación que integra el mapa y las funcionalidades comentadas en la sección anterior directamente en su smartphone. La aplicación para móviles de ANVM está dirigida a todo tipo de usuario que utilice smartphone y que además tenga algún tipo de relación con el complejo sobre el cuál se ha personalizado la aplicación.

1.4. Limitaciones

- **Limitaciones hardware:**
  - Se deben disponer de los elementos hardware necesarios para la construcción del robot de reconstrucción.
  - El sistema de reconstrucción ha de ejecutarse sobre un ordenador portátil puesto que se necesita que forme parte del carro durante la reconstruccción.
- **Limitaciones software:**
  - El programa de reconstrucción está compilado para Windows y se tiene que ejecutar en un ordenador con este sistema operativo.
  - Debe tener instaladas las librerías pertinentes de las que depende este proyecto para la correcta ejecución del mismo.

1.5. Supuestos y dependencias

Se asume que los requisitos que se describen en este documento son estables una vez que el equipo de desarrollo los ha aprobado. Cualquier cambio en ellos debe obtener la aprobación de todos los integrantes y será gestionado por los mismos.

Una dependencia que este proyecto debe tener en cuenta es la aplicación móvil sobre la que se mostrará el mapa que se va a visualizar. Después con la herramienta ANVM - Map Editor se rellenará toda la información que el cliente requiere, y por último se visualizará en la aplicación ANVM - Mobile App, en donde el usuario final pueda observar el mapa con su smartphone.

2. Requisitos específicos

Esta sección contiene todos los requisitos del software descritos con un nivel de detalle suficiente que permita:

- **A los desarrolladores:** diseñar un sistema acorde a estos requisitos.
- **A los probadores o testers:** realizar las pruebas necesarias para comprobar que se satisfacen estos requisitos.

En primer término se describen los requisitos que afectan a la funcionalidad, usabilidad, seguridad, rendimiento y compatibilidad. Después se definen las restricciones del diseño, los componentes adquiridos para el desarrollo y funcionamiento del sistema. Por último se mencionan las formas de contacto y soporte online disponibles para los usuarios, así como los requisitos de licencia, términos legales y copyright del presente producto.

2.1. Funcionalidad

Estos son los requisitos funcionales del proyecto ANVM. Se realiza una diferenciación entre las empresas u organizaciones y los usuarios finales. Las primeras, son los clientes que solicitan la herramienta en primer término y para cuyas necesidades se adapta el sistema. Los segundos, son los usuarios finales de la aplicación, aquellas personas que tienen algún tipo de relación con el entorno virtualizado y utilizan la herramienta que previamente ha sido personalizada.

Requisitos funcionales para empresas y PYMES

1. El sistema debe proporcionar una virtualización del mapa adecuada con suficiente nivel de detalle al cliente para el cuál se está personalizando el software. El sistema debe adaptarse a todos los tipos de lugares que el cliente necesite virtualizar.

2. La herramienta para añadir la información al mapa virtualizado debe ser capaz de cargar el mapa y debe ofrecer la posibilidad de personalización para adaptarse a cualquier tipo de cliente perteneciente a cualquier sector. De esta manera tanto una empresa pública de servicios como una empresa privada que regenta un centro comercial puedan encontrar las opciones de personalización adecuadas a su problema.

Requisitos funcionales para el usuario final

1. El mapa que virtualizado que se muestre en la aplicación móvil final ha de tener la suficiente calidad como para que la experiencia del usuario no se vea mermada por este aspecto.

2. El sistema debe poder localizar al usuario en el mapa virtualizado mediante el escaneo de un código QR. Estos códigos serán distribuidos en forma de pegatinas por todo el entorno. Cada uno de ellos alberga la información necesaria para la localización que es empleada por la aplicación móvil para conocer la posición exacta del usuario.

3. Debe ser capaz de cambiar el idioma de presentación de los menús si el usuario lo solicita desde la pantalla de inicio.

4. Cuando el usuario solicita conocer la ruta para desplazarse a algún lugar, dicha ruta debe ser óptima en distancia y debe mostrarse sobrescrita en el mapa del entorno.

5. El usuario debe poder realizar una visita virtual por el complejo mediante la navegación por los diferentes panoramas disponibles.

2.2. Usabilidad

Este proyecto esta pensado para ser utilizado por técnicos de ANVM. Estos son cualquiera de los desarrolladores, o cualquier persona familiarizada con el sistema de reconstrucción. Es necesario para la captura de datos del proceso, disponer de alguien que conozca el terreno que se desea virtualizar para conocer los elementos que deben pertenecer al futuro mapa que se va a construir. El sistema será por tanto utilizado por personal de ANVM que sepa controlar el robot, y por algún conocedor del espacio a reconstruir. Es sencillo aprender a manejar el carro por lo que cualquier persona podría aprender a utilizarlo siguiendo las directrices de los desarrolladores.

El mapa generado debe ser intuitivo y fiel a la realidad para que el usuario final pueda utilizar la aplicación móvil sin problemas, ni diferencias con el mapa real del recinto. Se debe tener en cuenta que un mapa mal generado puede conducir al usuario final a errores en la utilización del sistema, como por ejemplo perderse en el recinto.

2.3. Seguridad

En esta sección los requisitos que afectan a la seguridad del sistema de reconstrucción de ANVM. Se enumerarán los aspectos más importantes y se comentará brevemente en qué pueden afectar y cómo solucionarlos.

- El sistema debe guardar el progreso en la captura de datos. Si se produce un error inesperado durante la captura de datos, podemos haber perdido todo lo capturado hasta el momento y eso puede resultar muy molesto. Por tanto se debe guardar periódicamente los avances en la captura de datos.
- El dispositivo que captura los datos debe estar perfectamente montado. No se puede permitir que durante la captura se suelte alguna pieza, se salga una rueda, o cualquier situación similar, puesto que se deberá interrumpir la captura, y los datos pueden quedar corruptos o erróneos.

2.4. Rendimiento

Estos son algunos requisitos de rendimiento para determinadas funcionalidades de la aplicación en las que se requiere cierta cantidad de cálculos. El sistema de reconstrucción necesita de un ordenador con una capacidad de cómputo capaz de soportar varios threads leyendo constantemente datos de diferentes puertos USB. No se deben retrasar nunca demasiado estas lecturas puesto que pueden resultar en datos erróneos que no se corresponde con la realidad. Es decir que puede que los datos referentes a la posición e imagen no coincidan con lo que se está visualizando en ese instante, sino que sean de un instante posterior. En la sección de pruebas se muestran pruebas de rendimiento y todas son satisfactorias.

2.5. Restricciones del diseño

En esta sección se va a tratar cada una de las restricciones del diseño que imponen los lenguajes de programación, librerías de clases y otro software que se ha utilizado en el sistema. Puesto que se utilizan ejecutables de la librería MRPT la aplicación se debe ejecutar en varios pasos, por un lado la captura de datos y por otro la reconstrucción. Se ha de ejecutar en varios threads puesto que la captura de datos tiene que realizar lecturas constantemente por USB y de esta manera no se detienen el resto de procesos.

Para el diseño y construcción del robot se ha de tener cuenta el límite de presupuesto y los elementos disponibles. No se podrán utilizar más elementos que los que se listan en la siguiente sección. Esto puede limitar la calidad con la que se construyen los mapas en este proyecto. No obstante se podrán insertar mejoras y adquirir nuevos componentes siempre dentro del presupuesto.

2.6. Componentes adquiridos

Para el desarrollo de ANVM han sido necesarios los siguientes componentes:

- **Sensor Kinect de Microsoft:** instalado en el robot empleado para la virtualización. Utilizado para la toma de datos. Valorado en 115€.
- **Placa Arduino:** instalada en el robot empleado para la virtualización. Necesaria para la comunicación con el ordenador cuando se están recogiendo los datos en una virtualización. Valorada en 22€.
- **Encoders:** instalados en el robot empleado para la virtualización. Utilizados para calcular la distancia recorrida por el robot. Valorados en 8€.
- **Muelles especiales:** instalados en el robot empleado para la virtualización. Necesarios para el sistema de giro de las ruedas. Valorados en 5€.
- **Carro portátil:** empleado para sostener el arduino junto con el Kinect y los encoders instalados en las ruedas. Reutilizado, coste cero.
- **Sony Xperia S:** empleado para ejecutar las pruebas durante el desarrollo de la aplicación móvil. Valorado en 415€.
- **Samsung Galaxy R:** empleado para ejecutar las pruebas durante el desarrollo de la aplicación móvil. Valorado en 300€.
- **Sony Xperia U:** empleado para ejecutar las pruebas durante el desarrollo de la aplicación móvil. Valorado en 160€.

Capítulo 5

Plan de desarrollo

En esta sección de la memoria, se ofrece una visión clara de cada una de las fases que han tenido lugar durante el desarrollo del proyecto. Se hablará de cómo se han planificado dichas fases y del cumplimiento de los objetivos dentro de cada una de ellas. Con esta sección se pretende desengranar el proyecto en cada una de sus fases de investigación, implementación y pruebas y analizar cada una de las entregas que se han llevado a cabo para cada fase.

Se describe también la planificación de trabajo personal de cada uno de los desarrolladores del proyecto para cumplir los objetivos de cada fase, así como la colaboración de personajes externos al proyecto que ayudaron o asesoraron en algun momento del proyecto.

Cabe destacar que gracias a esta planificación mediante hitos a corto medio plazo, y la preparación de diversas entregas a lo largo del proyecto se han podido cumplir con los objetivos que se marcaron dentro del alcance del proyecto, y construir un proyecto sólido, estable, y fácilmente ampliable para cumplir el resto de objetivos que se quedaron fuera del alcance por falta de tiempo.

Todos los términos científicos, tecnicismos, siglas y demás expresiones que necesiten una breve explicación complementaria están recogidos en la sección Glosario.

1. Descripción del proyecto

Dentro de esta sección se proporciona una descripción de la finalidad y objetivos del proyecto. Se describe también el alcance previsto para cada una de las fases así como dol proyecto final. También se definen cada una de las entregas que se han ido cumpliendo dentro del proyecto junto con las susposiciones y limitaciones necesarias para cada entrega.

1.1. Propósito del proyecto, alcance y objetivos

El objetivo principal del proyecto consiste en la localización del usuario en interiores mediante un sistema de códigos QR, esta decisión se tomó después de varias semanas de investigación con otras tecnologías de reconocimiento de puntos en el espacio utilizando la cámara del Kinect. Esta idea se descartó rápidamente por la gran dificultad que conllevaba hacerlo de esta forma y el poco beneficio que se obtiene, ya que con un simple código QR es posible localizar al usuario, aunque no con la libertad que otorga esta otra tecnología.

Tras elegir el código QR como opción, se consigue situar al usuario en un mapa 2D generado mediante el proyecto desarrollado en paralelo con este, ver todos los puntos de interés del lugar en el que se encuentre o bien recibir indicaciones para dirigirse a

uno de ellos. A continuación se muestra los objetivos de cada una de los subproyectos que engloban ANVM.

El proyecto contiene una aplicación móvil desarrollada en Android, ANVM - Mobile App, cuyos principales objetivos que se plantearon son:

- Visualización e interacción con el mapa 2D.
- Escaneo y localización mediante códigos QR.
- Visualización de un listado de puntos de interés.
- Guiado del usuario a través del mapa mediante rutas a los puntos de interés.

Una aplicación para crear toda la base de datos y las localizaciones de los puntos de interés en el mapa 2D desarrollada en Java, ANVM - MapEditor, cuyos hitos son:

- Marcado y edición de los puntos de interés de cada mapa.
- Cálculo de distancias entre los puntos de interés.
- Cálculo del grafo de búsqueda y de las rutas óptimas entre los puntos de interés.
- Generación de ficheros de datos para la aplicación móvil con toda la información editada.

Un robot que sea capaz de obtener dicho mapa utilizando diversas tecnologías, ANVM - Virtual Mapping, cuyos objetivos son:

- Diseño y construcción de un robot capaz de calcular de forma autónoma su posición mientras se está moviendo.
- Diseño e implementación de un programa capaz de unir la información de la posición del robot, con las imágenes que se van capturando durante el recorrido.
- Generar un mapa del espacio interior a partir de toda la información recogida con el robot.

Para una descripción más detallada del proyecto se recomienda acudir a la sección Visión.

1.2. Suposiciones y limitaciones

A continuación se enumeran cada una de las suposiciones y limitaciones a la hora de planificar cada uno de las fases dentro de la realización de este proyecto:

- Los desarrolladores del Proyecto ANVM actualmente también trabajan a tiempo parcial o completo en sus respectivas empresas, y por ello, la planificación fue adaptada al horario de cada uno.
- Además se debe tener en cuenta que el desarrollo en la plataforma Android es completamente desconocida para los desarrolladores pues supone una gran limitación. También se ha realizado un fuerte estudio de las posibilidades de localización mediante códigos QR frente a otras opciones, y la técnica de tilear imágenes como mejor candidato a la hora de reducir el procesamiento gráfico de los terminales y su correcta utilización.

- La investigación sobre las diversas tecnologías que se aplican para la reconstrucción de los mapas 2D ha condicionado en gran medida tanto los objetivos, como la planificación dentro cada fase del proyecto, puesto que no se tenía ningún conocimiento previo sobre dichas tecnologías antes de la realización del proyecto.

- Las inversiones de tiempo no pudieron ser constantes en el tiempo puesto que cada desarrollador del equipo, ha dedicado el tiempo que podía dentro de sus obligaciones dentro de la carrera (otras asignaturas, exámenes y prácticas) por lo que su dedicación puede descender en épocas de exámenes, o ascender en otras. Como mínimo se calcula que cada desarrollador invirtió unas 14 horas semanales de media hasta la entrega de la versión final.

- Este proyecto deberá entregarse el día 7 de Junio, por lo que deberá estar terminado para entonces dejando de lado todos los objetivos que queden fuera del alcance.

1.3. Entregas del proyecto

En el cuadro 7 aparecen las fechas aproximadas de las entregas que se programaron como objetivo a lo largo de todo el desarrollo del proyecto:

Fecha	Producto
24/05/2012	Presentación de la idea del proyecto a la convocatoria de Wayra 2012.
19/11/2012	Primera versión de la aplicación móvil. Primera versión del programa de reconstrucción.
17/12/2012	Primera versión del editor de mapas y primera versión de la documentación.
15/01/2013	Segunda versión de la aplicación móvil. Primera versión del prototipo robot para las reconstrucciones.
18/02/2013	Segunda versión de la documentación. Finalizado el editor de mapas. Primera reconstrucción realizada por el robot.
25/03/2013	Coloreado del mapa dentro de la aplicación móvil mostrando todos los elementos del mismo, así como la posibilidad de mostrar rutas entre dos puntos. Implementación de mejoras en el robot y reconstrucciónes mayores y más sólidas.
07/06/2013	Última versión de la documentación para ser entregada. Finalizada la aplicación móvil con las funcionalidades mencionadas anteriormente. Se añade la funcionalidad de vista a través de panoramas. Última versión del programa de reconstrucción con interfaz y corrección de errores.

Tabla 7: Entregas del proyecto

1.4. Evolución del plan de desarrollo de software

El Plan de Desarrollo del Software ha ido modificándose a lo largo del proceso de investigación comenzado en Agosto de 2012 debido, principalmente, a las limitaciones que se imponían cada vez que se investigaba un nuevo software para la reconstrucción de la superficie, proceso de investigación detalladamente explicado en el proyecto ANVM - Virtual Mapping. A partir de finales de Noviembre, no hubo ningún cambio significativo en esta planificación, a excepción de pequeñas modificaciones del alcance del proyecto, a fin de evitar o resolver algunos riesgos previstos, o de la implementación de herramientas auxiliares.

Plan de proyecto inicial

Este plan fue desarrollado durante la primera fase del proyecto, en Agosto de 2012. Los objetivos del proyecto en el momento diferían mucho de los del plan actual, sin embargo, algunos aspectos han permanecido intactos.

1. Primer prototipo
   a) Generar una nube de puntos utilizando la cámara Kinect.
   b) Ser capaces de situarnos en el espacio 3D mediante fotos.
   c) Prototipo de aplicación móvil.
2. Segundo prototipo
   a) Guardar de forma eficiente la nube de puntos generada por el Kinect en la nube.
   b) Localización mediante video en el espacio 3D.
   c) Segundo prototipo móvil con mayor funcionalidad.
3. Entrega final
   a) Aplicación móvil completa capaz de:
      - Localizar al usuario en el espacio 3D mediante vídeo.
      - Buscar puntos de interés.
      - Consultar el mapa desde la pantalla del smartphone.
      - Obtener la ruta óptima para ir de un punto de interés a otro.
      - Consultar la recreaciones panorámicas de los puntos de interés.

Durante la generación del primer prototipo, se encontraron multitud de limitaciones en la localización mediante fotos o vídeos en la nube de puntos 3D. Por esta razón, se descartó este plan de fase salvo el diseño de la aplicación móvil. En la sección de investigación de la memoria del proyecto ANVM - Virtual Mapping están descritas todas las dificultades encontradas en el proceso y las conclusiones sacadas al respecto.

Segundo plan de proyecto

El Segundo Plan de Proyecto se enfocó en una nueva forma de localización, mucho más sencilla y menos costosa, la utilización de códigos QR. Con esta nueva idea, se descarga la información almacenada en la nube al no tener que guardar toda la nube de puntos, a parte de evitar procesos complejos de obtención de puntos clave de las capturas hechas con el móvil, reduciendo el coste computacional de la localización.

1. Primer prototipo

   a) Generar un mapa 2D empleando el Kinect y guardar los datos.

   b) Primer prototipo de aplicación móvil con interfaz simple.

2. Segundo prototipo

   a) Construir un robot que permita grabar con el menor error posible.

   b) Primera versión del programa de reconstrucción capaz de comunicarse con el robot.

   c) Aplicación móvil con una interfaz mejorada y con pequeñas funcionalidades, lector QR, manejo de mapas y listado de puntos de interés.

   d) Aplicación Java que permita generar la información sobre los puntos de interés en un mapa, distancias, localización y descripción.

3. Entrega final

   a) Robot completamente funcional que permita obtener un mapa 2D de una superficie obteniendo la información con el Kinect y la odometría.

   b) Programa de reconstrucción que ofrezca total integración con el robot anterior y permita generar los mapas.

   c) Aplicación Java que sea capaz de generar la información sobre los puntos de interés en un mapa (distancias, localización o descripción entre otras), el grafo de búsqueda y las rutas óptimas entre los puntos de interés, y la creación automática de códigos QR para cada nodo del grafo.

   d) Aplicación móvil completa capaz de:

   - Localizar al usuario en el espacio 3D mediante el reconocimiento de códigos QR.
   - Buscar puntos de interés.
   - Consultar el mapa desde la pantalla del smartphone.
   - Obtener la ruta óptima para ir de un punto de interés a otro.
   - Consultar la recreaciones panorámicas de los puntos de interés.

2. Organización del proyecto

En esta sección se va explicar en detalle la organización del proyecto ANVM en conjunto, haciendo referencias a los dos proyectos paralelos. En primer lugar se trata la estructura organizativa de manera general, se mencionan los colaboradores externos y se enumera los roles de cada uno de los integrantes del equipo de desarrollo de ANVM.

2.1. Estructura organizativa

Como se ha comentado anteriormente, ANVM está formado por dos equipos de desarrollo que en conjunto han desarrollado las herramientas de ANVM. Estos dos proyectos son ANVM - Virtual Mapping y ANVM - Mobile App. El primero de ellos engloba toda la parte de la virtualización de entornos interiores mediante la captación de imágenes y datos por parte de un robot construido por los propios desarrolladores. El segundo es el encargado de la aplicación móvil de Android. A pesar de que cada equipo consta de dos desarrolladores dedicados a campos distintos, se puede asegurar que cada uno se ha visto involucrado en mayor o menor medida en todas las tareas que conforman este proyecto. En la figura 32 podemos ver un cuadro explicativo de como se organiza el proyecto.

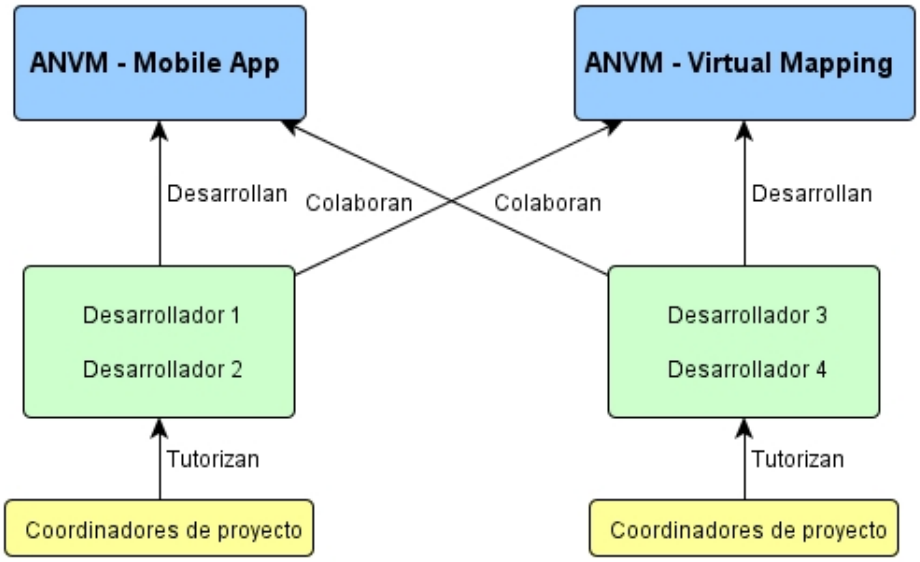

Figura 32: Organización ANVM

2.2. Colaboradores externos

A continuación se van a enumerar una serie de colaboradores que no son los integrantes del equipo del proyecto, pero que han ayudado en mayor o menor medida a que el proyecto saliera adelante.

- **Mr Changchang Wu** miembro del Departamento de Informática de la Universidad de Carolina del Norte y cuyas investigaciones están centradas en las reconstrucciones 3D. Ha colaborado con los desarrolladores de ANVM prestando su experto criterio y experiencia (vía correo electrónico) en la conversión de puntos SIFT, necesario para las primeras investigaciones relativas a la localización a través de imágenes.
- **Don José Luis Blanco** profesor de la Universidad de Málaga, y desarrollador de la librería Mobile Robot Programming Tool colaboró mediante una serie de correos y mensajes en el foro de su propia librería aclarando ciertas dudas de los integrantes del equipo a la hora de utilizarla.
- **Doña María Laura Hernando Guadaño** profesora de Ingeniería Aeronáutica de la Universidad Politécnica de Madrid, traductora e intérprete jurada de Inglés y Francés. Ha colaborado en la traducción de todas y cada una de las palabras y frases que aparecen en la aplicación Android en los idiomas francés e inglés.
- Familiares de los integrantes del proyecto colaboraron en las labores de reconstrucción de interiores facilitando la realización de pruebas necesarias para probar la eficacia y buen funcionamiento del proyecto, entre los que nombramos a José Peñalver y Antonia Santorio.
- ANVM es un proyecto de fin de carrera, y como tal, está tutorizado por profesores de la Universidad Complutense de Madrid. Estos profesores no han participado directamente en el desarrollo e implementación del proyecto pero sí que han asesorado y tutorizado a los alumnos durante la elaboración del mismo. Además, han proporcionado herramientas técnicas necesarias para el proyecto y han facilitado la utilización de aulas y laboratorios para las reuniones.

2.3. Roles y responsabilidades

En esta sección vamos a presentar los roles y responsabilidades de los miembros del equipo de deasarrollo de ANVM. Es importante destacar que cada desarrollador ha estado implicado en todos los ámbitos del proyecto y las responsabilidades que aquí se mencionan no han sido tarea únicamente de la persona que aquí aparece. Estos son las personas que han trabajado y ayudado al desarrollo de ANVM.

- **Miguel Gutiérrez García-Cuevas**
  - Rol: desarrollador principal de ANVM - Mobile App y colaborador de ANVM - Virtual Mapping.
  - Responsabilidad principal: encargado del desarrollo de la aplicación en Java, ANVM - MapEditor, para el guardado de información y decoración de los mapas.
- **Víctor Ortiz García**
  - Rol: desarrollador principal de ANVM - Mobile App y colaborador de ANVM - Virtual Mapping.

- Responsabilidad principal: supervisor de la memoria del proyecto ANVM - Mobile App y encargado de la aplicación de Android de ANVM.

- **Alejandro Peñalver Santorio**

  - Rol: desarrollador principal de ANVM - Virtual Mapping y colaborador en ANVM - Mobile App.

  - Responsabilidade principal: supervisor de la memoria del proyecto ANVM - Virtual Mapping y encargado de la toma de datos con el robot y su posterior procesamiento.

- **Ricardo Pragnell Valentín**

  - Rol: desarrollador principal de ANVM - Virtual Mapping y colaborador en ANVM - Mobile App.

  - Responsabilidade principal: encargado de la construcción, manejo y mantenimiento del robot para la toma de datos.

- **Juan Carlos Fabero Jiménez**

  - Rol: director del proyecto ANVM - Virtual Mapping.

  - Responsabilidades: asesorar y tutorizar a los alumnos desarrolladores de ANVM. Más concretamente, en el campo del apartado hardware del proyecto, lo que incluye el Kinect, el arduino, la construcción del robot y la toma de datos por parte del mismo.

- **Guadalupe Miñana Ropero**

  - Rol: ayudante de dirección del proyecto ANVM - Virtual Mapping.

  - Responsabilidades: asesorar y tutorizar a los alumnos desarrolladores de ANVM. Cabe destacar en su función la promoción del proyecto más allá de la UCM en diferentes procesos de incubadoras de ideas.

- **Luis Garmendia Salvador**

  - Rol: director del proyecto ANVM - Mobile App.

  - Responsabilidades: asesorar y tutorizar a los alumnos desarrolladores de ANVM.

- **María Victoria López López**

  - Rol: ayudante de dirección del proyecto ANVM - Mobile App.

  - Responsabilidades: asesorar y tutorizar a los alumnos desarrolladores de ANVM. Más concretamente, dedicada al posicionamiento del producto más allá de un trabajo de fin de carrera y supervisando la documentación generada a lo largo del desarrollo.

3. Proceso de gestión

Dentro de esta sección se detallará cómo ha sido gestionado el proyecto, cómo se ha planificado, cómo se han dividido las responsabilidades, y cómo se han organizado las diferentes iteraciones que componen el proceso de desarrollo software.

3.1. Estimaciones

La entrega final del proyecto está programada para el día 7 Junio de 2013. Podemos estimar una cantidad de horas globales dedicadas a todo el proyecto mediante una simple multiplicación: Hay 2 miembros en el proyecto, cada uno de ellos ha trabajado un promedio de 14 horas semanales. El proyecto comenzó su fase de investigación en Agosto del año 2012, hasta su finalización el día 7 de Junio de 2013, habiendo transcurrido 311 días, es decir, 44,5 semanas. Por lo tanto, se estima que el proyecto tendrá una cantidad de horas igual a 623 por persona, 1246 por equipo o 2492 entre todos los desarrolladores del proyecto ANVM.

3.2. Plan de proyecto

A continuación en esta sección se va a presentar el plan general con el que se ha trabajado en el proyecto ANVM. Se describen en cada una de las subsecciones siguientes cada de los aspectos de este plan general de proyecto.

Plan de fase

El proyecto ANVM fue planificado en 3 fases claramente diferenciadas que permitieron a los desarrolladores dividir cómodamente el trabajo y planificar bien cada una de las tres entregas y tenerlas preparadas más o menos dentro de las fechas previstas. A continuación se presenta un resumen de cada una de las fases, junto con un diagrama de grant por cada una que representa las principales tareas a llevar a cabo dentro de dicha fase.

Fase de investigación

En esta primera fase del proyecto es donde se lleva a cabo la mayor parte del trabajo de investigación del proyecto. Por tanto, es en esta fase en la que se descubrieron ciertos aspectos que hicieron modificar algunos de los objetivos del resto de las fases, que venían condicionados por el trabajo de investigación previo. En la figura 33 se puede ver un diagrama de Gantt donde se planificaron a grandes rasgos cada uno de los campos que era necesario investigar, y cuánto tiempo habría que dedicar a cada campo.

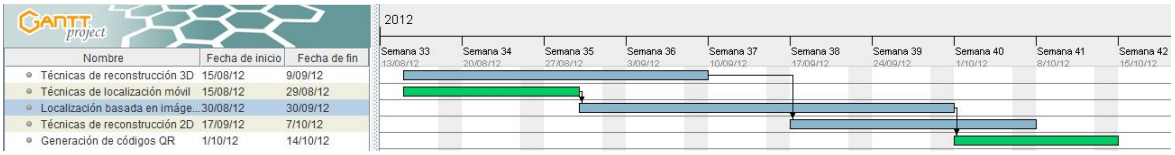

Figura 33: Diagrama de Gantt fase de investigación

Este diagrama fue ajustándose según las investigaciones llevadas a cabo ya que alguna tarea, como la localización basada en imágenes, tomó más tiempo del que se había previsto. Así mismo, los resultados de ciertas investigaciones motivaron nuevas tareas (explorar nuevos campos de investigación) como la generación de códigos QR, y las técnicas de reconstrucción 2D.

Primer prototipo

Una vez completada la fase de investigación, y con sus resultados en la mano, se acordaron los objetivos finales para el primer prototipo del proyecto. Se realizó una planificación de cada una de las tareas a llevar a cabo para cumplir cada objetivo (ver figura 34) y excepto en ciertas tareas, la planificación fue respetada y los objetivos cumplidos a tiempo. En color caqui se representan las tareas de diseño, en azul y verde las de implementación, y en granate las de construcción y pruebas con el hardware del proyecto.

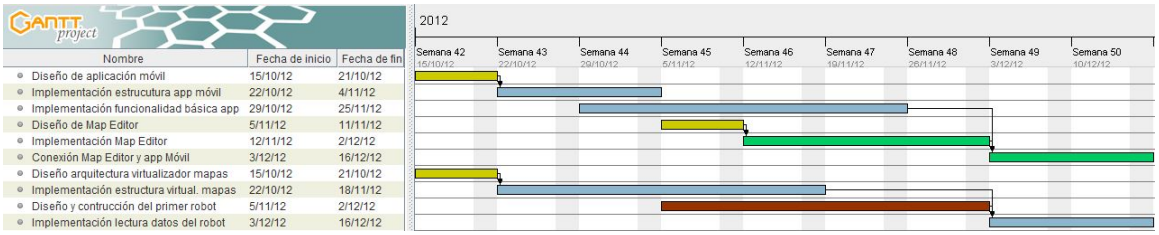

Figura 34: Diagrama de Gantt primer prototipo

Cabe destacar que en esta planificación la mayoría de las tareas no se podían solapar con las siguientes puesto que dependían unas de otras, y por tanto deberían terminarse a tiempo porque sino se retrasaría toda la planificación. El equipo tuvo muy en cuenta esto y si alguien terminaba su tarea antes de tiempo, ayudaba a quien fuera más retrasado con su tarea. Por esto la mayoría de las tareas se realizaron más o menos para el día indicado en el diagrama, a excepción de la última tarea (pruebas sobre la virtualización) que se tuvo que posponer una semana para terminar la implementación.

Segundo prototipo

Tras el periodo vacacional de navidades se desarrolló el plan de fase para el segundo prototipo, una versión “final” del proyecto a falta de realización de pruebas que quedarían para la última parte del curso. A continuación se muestra el último diagrama de Gantt que se hizo para este proyecto con una leyenda similar al anterior.

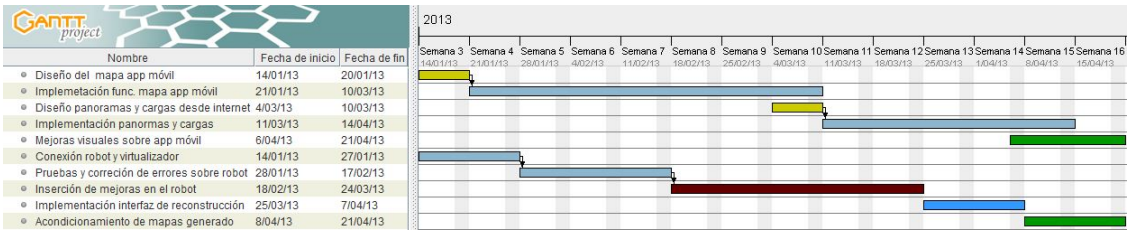

Figura 35: Diagrama de Gantt segunto prototipo

Cabe destacar que durante el periodo de exámenes no se podía trabajar todas las horas deseadas en el proyecto, y por eso se extendió la planificación en esas fechas para dar posibilidad a los desarrolladores de realizar sus exámenes. Aunque se demoraron algunas tareas, se llegó al objetivo de tener un segundo prototipo con mayor funcionalidad para antes de mayo.

Entrega final

Para esta última fase no se realizó ninguna planificación exhaustiva ya que el único cometido era realizar pruebas sobre el trabajo realizado y arreglar los errores que iban surgiendo así como terminado los puntos de la documentación que faltaban. Esta última fase comprende el mes de mayo hasta el final de la entrega el 7 de Junio.

Versiones

De acuerdo con el plan de fase, se planifican 3 versiones con una funcionalidad bien definida y diferenciada para cada una de manera incremental:

- Primer prototipo: 17/12/2012

  Para esta primera versión se propone tener diseñada e implementada una estructura para la aplicación móvil lo suficientemente estable y versátil como para poder implementar sobre ella el resto de funcionalidad deseada más adelante. Se implementa también en esta versión la transición entre cada una de las Activities o pantallas de la aplicación móvil, aún sin funcionalidad.7

  Por otra parte se diseña e implementa por completo la aplicación Map Editor que permita la conexión entre la aplicación móvil y los mapas generados por la virtualización.

  Para la parte de virtualización se planea tener construido una primera versión del robot que realiza las virtualizaciones, así como una primera versión del programa de reconstrucción y comunicación con el robot

- Segundo prototipo: 1/05/2013

  En la segunda versión se implementa toda la funcionalidad planificada para este proyecto y descrita en secciones anteriores. Se finaliza la conexión entre Map Editor y la aplicación móvil, así como una versión mejorada del robot capaz de realizar virtualizaciones de espacios mayores y con mayor precisión.

- Prototipo final: 7/06/2013

  Esta versión es la primera versión estable del proyecto ANVM que no incluye nueva funcionalidad desde el segundo prototipo. Se plantean simplemente ciertas mejoras y arreglo de errores con respecto a la versión anterior.

Calendario del proyecto

Hito	Periodo de realización	Fase
Investigación técnicas de reconstrucción	15/08/2012 - 08/10/2012	Investigación
Investigación tecnologías de localización	15/08/2012 - 01/10/2012	Investigación
Investigación tecnologías móvil	15/09/2012 - 15/10/2012	Investigación
Implementación estructura de aplicación móvil	15/10/2012 - 25/11/2012	Primer prototipo
Implementación Map Editor	15/11/2012 - 17/12/2012	Primer prototipo
Construcción del robot para reconstrucción	29/10/2012 - 19/11/2012	Primer prototipo
Implementación programa de reconstrucción	22/10/2012 - 01/12/2012	Primer prototipo
Implementación funcionalidad de la aplicación móvil	14/01/2013 - 08/04/2013	Segundo prototipo
Aumento de la capacidad de reconstrucción del robot	14/01/2013 - 04/03/2013	Segundo prototipo
Mejoras sobre el sistema de reconstrucción	04/03/2013 - 22/04/2013	Segundo prototipo
Acondicionamiento de la aplicación móvil	08/04/2013 - 07/06/2013	Versión final
Acondicionamiento de mapas reconstruidos	15/04/2013 - 07/06/2013	Versión final

Tabla 8: Calendario del proyecto ANVM

Plan de formación

En el proyecto ANVM se utilizan ténicas y tecnologías muy novedosas y por lo tanto el equipo desarrollador necesita cierta formación y práctica con dichas tecnologías. Se realizó un plan de formación en dichas tecnologías durante la fase de investigación, a medida que se fueron investigando las tecnologías que se iban a utilizar, se realizaron prácticas sobre las mismas mediante tutoriales y ejemplos que ayudaron a la comprensión y posterior utilización de las mismas.

Por tanto los periodos de formación coinciden con los de investigación. En la tabla 9 se presenta a modo de resumen las tecnologías sobre las que se tuvo que formar el equipo

y el periodo aproximado sobre el que se planificaron las mismas mientras se investigaba sobre ellas.

Tecnología	Periodo de formación
Dispositivo Kinect	15/08/2012 - 22/08/2012
Android OS	01/10/2012 - 15/10/2012
Reconstrucción 3D	01/09/2012 - 10/09/2012
Reconstrucción 2D	01/10/2012 - 15/10/2012
Arduino robot	22/10/2012 - 29/10/2012

Tabla 9: Plan de formación ANVM

Presupuesto

A pesar de que la mayoría de los dispostivos necesarios para la realización del proyecto ya estaban a disposición del equipo de desarrolladores antes de comenzar el proyecto, algunos hubo que adquirirlos posteriormente para completar el proyecto. Puesto que hubo que construir un robot, fue necesario comprar ciertos componentes hardware para construirlo. A continuación se presentan los elementos que se utilizaron en el proyecto, tanto los que hubo que adquirir, como de los que ya se disponía.

- Elementos de los que se disponía al inicio del proyecto:
  - **Sensor Kinect de Microsoft:** instalado en el robot empleado para la virtualización. Utilizado para la toma de datos. Coste 115€
  - **Sony Xperia S:** empleado para ejecutar las pruebas durante el desarrollo de la aplicación móvil. Coste 415€.
  - **Samsung Galaxy R:** empleado para ejecutar las pruebas durante el desarrollo de la aplicación móvil. Coste 300€.
  - **Sony Xperia U:** empleado para ejecutar las pruebas durante el desarrollo de la aplicación móvil. Coste 160€.
  - **Carro portátil:** empleado para sostener el Arduino junto con el Kinect y los encoders instalados en las ruedas. Reutilizado, coste cero
  - **Piezas de lego:** necesarias para montar los encoders sobre las ruedas del robot.
- Elementos que se tuvieron que adquirir con posterioridad:
  - **Placa Arduino Uno:** instalada en el robot empleado para la virtualización. Necesaria para la comunicación con el ordenador cuando se están recogiendo los datos en una virtualización. Coste 22€.
  - **Encoders:** instalados en el robot empleado para la virtualización. Utilizados para calcular la distancia recorrida por el robot. Coste 8€.
  - **Medidor láser:** para realizar comprobaciones sobre el cálculo de distancias: 0€ (Prestado por colaboradores externos).

3.3. Planes de iteración

En esta sección se mostrarán todos los planes de iteración que se llevaron a cabo dentro de cada fase. Para cada cada uno de ellos se mostrarán los objetivos de la iteración, el periodo de realización, así como la planificación que se hizo del trabajo para cada miembro del equipo ANVM. En las tablas 10 y 11 se muestra un resumen de los objetivos y fechas de cada iteración, cuya planificación será posteriormente explicado en la correspondiente subsección.

Iteración	Periodo	Fase	Objetivos
Iteración 1	15/08/2012 - 17/09/2012	Investigación	- Investigación técnicas reconstrucción 3D - Localización móvil - Localización basada en imágenes: reconstrucción
Iteración 2	17/09/2012 - 15/10/2012	Investigación	- Técnicas de reconstrucción 2D - Localización basada en imágenes: localización - Android y códigos QR
Iteración 3	15/10/2012 - 12/11/2012	Primer prototipo	- Diseño de la estructura de la aplicación móvil - Implementación de la estructura de la aplicación móvil - Diseño del editor de mapas - Diseño e implementación de la estructura del programa de reconstrucción - Diseño del robot para reconstrucción
Iteración 4	12/11/2012 - 17/12/2012	Primer prototipo	- Implementar funcionalidad básica en aplicación móvil - Implementar editor de mapas - Conexión entre editor de mapas y aplicación móvil - Construcción del robot - Lectura de datos del robot - Conexión robot con programa de reconstrucción - Pruebas sobre el sistema de reconstrucción

Tabla 10: Iteraciones del proyecto ANVM (1 a 4)

Iteración	Periodo	Fase	Objetivos
Iteración 5	14/01/2013 - 04/03/2013	Segundo prototipo	- Funcionalidad en el mapa de la aplicación móvil - Corrección de errores en la aplicación móvil - Corrección de errores del sistema de reconstrucción - Ampliación de espacios de reconstrucción
Iteración 6	04/03/2013 - 15/04/2013	Segundo prototipo	- Implementación de navegación 3D (panoramas) - Implementación y creación de datos en otros idiomas - Desarrollo de la base de datos SQL - Ampliación de funcionalidad: mostrar puntos de interés - Mejoras en el sistema de reconstrucción - Implementación de la interfaz del sistema de reconstrucción
Iteración 7	15/04/2013 - 07/06/2013	Entrega final	- Mejoras visuales en la aplicación móvil - Alineamiento de los panoramas y hotspot - Ampliación de funcionalidad: botones leyenda y SOS - Realización de pruebas y corrección de errores en la aplicación móvil - Realización de pruebas y corrección de errores en el sistema de reconstrucción

Tabla 11: Iteraciones del proyecto ANVM (5 a 7)

Plan de iteración 1

La primera iteración del proyecto coincide con el inicio de la fase de investigación. Su duración es de 4 semanas en las que el equipo comienza a investigar sobre los temas hacia los que se desea enfocar el proyecto. La investigación durante esta iteración condiciona en gran parte la que se lleva a cabo en la siguiente iteración, puesto que los descubrimientos en estos campos terminan de fijar claramente los objetivos del proyecto. A continuación se enumeran y describen dichos objetivos, así como los integrantes del equipo a los que se les planificó la tarea.

1. **Investigación técnicas reconstrucción 3D**

   El proyecto surge con esta idea, y por tanto es el primer campo que se ha de investigar. Dentro de este objetivo se engloba todo lo referente al uso del Kinect en la reconstrucción de espacios tridimensionales y se planifica la investigación de ciertas librerías que ayudan en este campo. Para más información consultar la sección de investigación.

   Este objetivo fue asignado a Victor Ortiz, Miguel Gutiérrez, Ricardo Pragnell y Alejandro Peñalver

2. **Localización móvil**

   Este objetivo de investigación se comienza de forma paralela al anterior dado que es otro de los puntos importantes del proyecto. Se pretende estudiar las técnicas de localización que se pueden implementar sobre dispositivos móviles. El estudio de este campo acaba por introducir un objetivo que no estaba contemplado inicialmente, la localización basada en imágenes.

   Este objetivo fue asignado a Alejandro Peñalver y Miguel Gutiérrez.

3. **Localización basada en imágenes: reconstrucción**

   Se llega a la conclusión, tras la previa investigación en la localización móvil, de que ésta técnica aporta buenos resultados en la localización y se encuentran estudios que aplican esta técnica en dispositivos móviles. Se planifica por tanto para esta iteración la investigación de esta técnica pero tan sólo para la parte de reconstrucción a partir de imágenes. La localización propiamente dicha se deja para la siguiente iteración.

   Este objetivo fue asignado a Ricardo Pragnell y Victor Ortiz.

Plan de iteración 2

El segundo plan de iteración dura también 4 semanas destinadas a acabar la investigación necesaria del proyecto y dejar los objetivos y alcance totalmente especificado. Debido a que ciertas partes de la investigación eran bastante extensas se dividieron entre estas dos iteraciones y por tanto algunas de las tareas de esta iteración son continuación de una tarea de la iteración anterior. A continuación se describen los objetivos de esta iteración así como la asignación al grupo de trabajo correspondiente.

1. **Técnicas de reconstrucción 2D**

   Durante la anterior iteración se investigaron las técnicas para 3D y puesto que se descartaron del alcance del proyecto se decidió tomar este camino. Dentro de esta tarea se engloba la investigación de todas las técnicas que permitan reconstruir un cierto espacio interior y generar un mapa 2D. También se incluye la investigación y prueba de librerías que permitan utilizar estas técnicas de manera más cómoda y se ajusten a las necesidades del proyecto.

   Este objetivo fue asignado a Ricardo Pragnell y Alejandro Peñalver.

2. **Localización basada en imágenes: localización**

   Este objetivo comenzó en la iteración anterior, y durante esta iteración se plantea la investigación de la parte propia de la localización. Se pone como objetivo comprender esta técnica para su posible posterior utilización así como la ejecución de pruebas que permitan ver el error en el cálculo de la posición.

   Este objetivo fue asignado a Victor Ortiz, Miguel Gutiérrez, Ricardo Pragnell, Alejandro Peñalver.

3. **Android**

   Dado que la aplicación móvil se implementa en esta plataforma se planifica un estudio e investigación de dicho sistema operativo. Se realizan pequeñas aplicaciones de prueba que sirven a los integrantes del grupo para familiarizarse con el entorno de desarrollo Android.

   Este objetivo fue asignado a Victor Ortiz, Miguel Gutiérrez, Ricardo Pragnell y Alejandro Peñalver.

4. **Códigos QR**

   Se plantea este sistema para la localización en la aplicación móvil tras la investigación de otras técnicas. Para esta iteración se planifica una investigación sobre librerías que ayuden a su implementación en el móvil, así como otras que permitan generar los propios códigos.

   Este objetivo fue asignado a Victor Ortiz y Miguel Gutiérrez.

Plan de iteración 3

Este plan de iteración se planificó también para 4 semanas y corresponde con el primer periodo de diseño e implementación del proyecto. Se comienza a diseñar la arquitectura, y se pretende dotar a ANVM de una estructura sobre la que se pueda ampliar fácilmente funcionalidad en las próximas iteraciones. Se trata de una planificación bastante ambiciosa en el sentido de que hay muchas tareas que completar, sin embargo se planearon estas semanas de mayor trabajo porque el equipo desarrollador estaba bastante libre en esta época del curso. A continuación se describen los objetivos de esta iteración así como la distribución del trabajo entre los integrantes del equipo.

1. **Diseño de la aplicación móvil**

   Antes de implementar nada de la aplicación se planea dedicar un tiempo a estudiar bien qué pantallas (Activities) va a tener la app así como los elementos (botones, imágenes...) que debería tener cada una. Se pretende obtener un esbozo en papel de la transición entre activities que definan por completo la futura funcionalidad de la app.

   Puesto que en esta tarea se debían decidir muchas cuestiones importantes que influían en el resultado final del proyecto, se asignó esta tarea a todos los integrantes del equipo: Miguel Gutiérrez, Victor Ortiz, Ricardo Pragnell y Alejandro Peñalver.

2. **Implementación de la estructura de la aplicación móvil**

   Una vez completada la tarea 1 se planeó dar comienzo a esta tarea que consiste en realizar una implementación de la estructura de las activities diseñadas anteriormente. Se trata de realizar un esqueleto sobre el que implementar después la funcionalidad necesaria dentro de cada activity.

   Este objetivo fue asignado a Miguel Gutiérrez y Victor Ortiz.

3. **Diseño del editor de mapas**

   Se planifica esta tarea para esta iteración con el objetivo de tener todo lo listo para implementar en la siguiente iteración. Se trata de diseñar la arquitectura del Map Editor a desarrollar en java. La funcionalidad estaba bien definida y se dejó para el final de la iteración puesto que no tenía demasiada dificultad.

   Este objetivo fue asignado a Miguel Gutiérrez, Ricardo Pragnell y Alejandro Peñalver.

4. **Diseño e implementación de la estructura del programa de reconstrucción**

   Junto con el diseño e implementación de la aplicación móvil esta fue una de las tareas más críticas en esta iteración pues era necesario tener listo y preparado el programa para una vez construido el robot comenzar a hacer pruebas lo antes posible. Se plantea diseñar e implementar la estrucutura del programa así como la funcionalidad principal, para después incorporarle la información proveniente del robot.

   Este objetivo fue asignado a Ricardo Pragnell y Alejandro Peñalver.

5. **Diseño del robot para reconstrucción**

   Esta tarea se planifica en esta iteración para dejar preparado el diseño del robot y poder comprar los materiales y construirlo en la próxima iteración.

   Este objetivo fue asignado a Ricardo Pragnell y Alejandro Peñalver.

Plan de iteración 4

Esta iteración consta de 5 semanas dentro de las cuáles se pretente obtener un primer prototipo con cierta funcionalidad. Es por eso que se da prioridad a la implementación para obtener una primera versión del proyecto. Los objetivos de esta iteración son por tanto ambiciosos pero posibles y durante esta iteración el equipo se vuelca en el desarrollo. Se trata de conseguir una versión con funcionalidad básica, tanto para la aplicación móvil como para el robot, que permita durante las próximas iteraciones mejorarla y añadir funcionalidad.

1. **Implementación funcionalidad básica en la app móvil**

   Sobre la estructura y mínima funcionalidad montada en la anterior iteración, se continúa implementando aspectos básicos de la aplicación. Se plantean para esta iteración la correcta transición entre activities, búsqueda dentro de la aplicación, y captación de códigos QR.

Este objetivo fue asignado a Victor Ortiz y Miguel Gutiérrez.

2. **Implementación del editor de mapas**

   Una vez diseñada tanto la interfaz como el contenido de este programa se planifica su implementación para esta iteración. Se trata de finalizar por completo la implementación del editor, a falta de posibles mejoras o corrección de bugs posteriores, de manera que se pueda utilizar la información generada por éste para la aplicación móvil en próximas iteraciones.

   Este objetivo fue asignado a Miguel Gutiérrez, Ricardo Pragnell y Alejandro Peñalver.

3. **Conexión entre el editor de mapas y app móvil**

   Con el aspecto del fichero XML generado por el Map Editor ya bien definido, se planifica para el final de esta iteración comenzar a preparar la aplicación móvil para que reciba los datos generados por el Map Editor. Se plantea comenzar con esta conexión con la intención de acabarla si fuera necesario en la próxima iteración.

   Este objetivo fue asignado a Miguel Gutiérrez y Victor Ortiz.

4. **Construcción del robot**

   Se planifica para el comienzo de esta iteración construir el robot diseñado anteriormente capaz de aportar la información necesaria sobre la posición para poder realizar la reconstrucción. Se prioriza en este objetivo puesto que de él dependen las tareas 5, 6 y 7.

   Este objetivo fue asignado a Ricardo Pragnell y Alejandro Peñalver.

5. **Lectura de datos del robot**

   Este objetivo se planifica para después de terminar la construcción del robot. Se trata de realizar un programa capaz de realizar la lectura de datos mediante USB suministrados por robot Arduino.

   Este objetivo fue asignado a Ricardo Pragnell.

6. **Conexión robot con programa de reconstrucción**

   Se trata de adaptar los datos leídos desde el robot para conseguir hacer funcionar el sistema de reconstrucción. Una vez concluida esta tarea, estaría finalizada la primera versión del sistema de reconstrucción, a falta de la realización de pruebas que demuestren su funcionamiento.

   Este objetivo fue asignado a Alejandro Peñalver.

7. **Pruebas sobre el sistema de reconstrucción**

   Debido a utilizar hardware para la reconstrucción que puede no ser del todo exacto, se planificaron pruebas al final de la iteración para ajustar los parámetros que hicieran el mapa reconstruido lo más preciso posible.

   Este objetivo fue asignado a Alejandro Peñalver y Ricardo Pragnell.

Plan de iteración 5

Esta iteración se planifica con una duración de 7 semanas. Es la más larga del proyecto debido a que en medio de la misma se encuentra el periodo de exámenes de febrero, y se opta por dar más tiempo para la realización de las tareas dentro de esta iteración, de manera que los integrantes del equipo puedan dedicar parte de su tiempo a realizar los exámenes y prácticas propios de estas fechas. El trabajo principal de esta iteración consiste en cerrar todos los cabos sueltos que pudieran haber quedado de la iteración anterior, donde se realizó el primer prototipo, y comenzar a implementar nueva funcionalidad para el segundo prototipo. A continuación se muestra una lista de los objetivos de esta iteración así como la asignación del trabajo dentro del equipo.

1. **Funcionalidad sobre el mapa de la app móvil**

   El objetivo de esta tarea es implementar todo lo referente al mapa en la aplicación móvil. Se trata tanto de la visualización del mapa, como de la implementación del zoom y el desplazamiento por el propio mapa, con todo su contenido: puntos de interés, cálculo de rutas... (información proveniente del Map Editor).

   Este objetivo fue asignado a Miguel Gutiérrez y Victor Ortiz.

2. **Corrección de errores en la app movil**

   Esta tarea se planifica paralela a la anterior para ir arreglando todos los errores encontrados en el primer prototipo, así como los que se fueran encontrando mientras se implmentaba la funcionalidad del mapa en la aplicación.

   Este objetivo fue asignado a Miguel Gutiérrez y Victor Ortiz.

3. **Corrección de errores del sistema de reconstrucción**

   Tras las pruebas realizadas en la anterior iteración, se planifica para esta la realización de más pruebas para arreglar todos los errores que se encontraron y conseguir un sistema de reconstrucción sólido de cara al segundo prototipo.

   Este objetivo fue asignado a Alejandro Peñalver y Ricardo Pragnell.

4. **Ampliación de espacios de reconstrucción**

   Tras conseguir un sistema robusto se plantea como objetivo de esta iteración conseguir un sistema capaz de reconstruir espacios de mayor dimensión, puesto que para el primer prototipo se realizan pruebas en espacios pequeños donde el error acumulado es menor.

   Este objetivo fue asignado a Alejandro Peñalver y Ricardo Pragnell.

Plan de iteración 6

Esta iteración consta de 6 semanas. En ella se amplía nuevamente la funcionalidad de la aplicación móvil, como es la introducción de los panoramas o la implementación de una base de datos SQL. Para esta iteración se deja tanto tiempo porque se debe realizar la toma de fotografías para realizar los panoramas y las fotos de los puntos de interés, y porque se necesitan realizar pruebas para mejorar el robot de reconstrucción.

A continuación se describen los objetivos de esta iteración así como la distribución del trabajo entre los integrantes del equipo.

1. **Implementación de navegación 3D (panoramas)**

   Se realizan unas 400 fotografías y se emplea una herramienta para la creación de los panoramas, uniendo las imágenes entre sí. Por otro lado, se implementa la funcionalidad sobre la aplicación móvil que permite cargar un panorama asociado a cada punto de interés.

   Este objetivo fue asignado a Víctor Ortiz y Ricardo Pragnell.

2. **Implementación y creación de datos en otros idiomas**

   El objetivo de esta tarea es ofrecer al usuario diversos idiomas entre los que poder elegir para ver toda la aplicación, incluida la información sobre los puntos de interés (títulos o descripciones entre otros). Para ello se modifica la herramienta Map Editor para que genere dos ficheros, uno con la información del grafo de conexiones y las matrices de Floyd y otro con los puntos de interés, el cual será el que se modifique dependiendo del idioma.

   Este objetivo fue asignado a Víctor Ortiz y Miguel Gutiérrez.

3. **Desarrollo de la base de datos SQL**

   Esta tarea se planifica paralela a la anterior para ir preparando la base de datos, con todos los puntos de interés en los diferentes idiomas, sobre la que se realizan diferentes consultas para obtener puntos de interés, o realizar búsquedas por etiquetas entre otras opciones.

   Este objetivo fue asignado a Miguel Gutiérrez.

4. **Ampliación de funcionalidad: mostrar puntos de interés**

   Se planifica esta tarea para esta iteración con el objetivo de empezar a acabar toda la funcionalidad que la aplicación tiene.

   Este objetivo fue asignado a Miguel Gutiérrez.

5. **Mejoras en el sistema de reconstrucción**

   En esta tarea se plantea diseñar mejoras que ayuden a generar mejores mapas y mas sólidos. Esto influye tanto a mejoras técnicas en el robot, como a ajustes en el programa de reconstrucción y en la configuración de los algoritmos que ahí se utilizan.

   Este objetivo fue asignado a Alejandro Peñalver y Ricardo Pragnell.

6. **Implementación de la interfaz del sistema de reconstrucción**

   Esta tarea se planifica para dar una apariencia más amigable al sistema de reconstrucción mediante una interfaz que automatice el proceso de reconstrucción y haga más sencilla su ejecución.

   Este objetivo fue asignado a Alejandro Peñalver y Ricardo Pragnell.

Plan de iteración 7

Durante esta iteración se planean sobre todo tareas de corrección de errores y de introducción de pequeñas mejoras tanto en la aplicación móvil como en el sistema de reconstrucción.

1. **Mejoras visuales en la aplicación móvil**

   El objetivo de esta tarea es diseñar las interfaces finales para que la aplicación tenga una apariencia profesional y agradable para el usuario. Para realizar esta tarea se obtuvo varias opiniones de familiares, profesores y compañeros, así como un estudio de diferentes aplicaciones profesionales, tanto competidoras como no.

   Este objetivo fue asignado a Miguel Gutiérrez y Víctor Ortiz.

2. **Alineamiento de los panoramas y hotspot**

   Sobre los panoramas creados en la iteración anterior, se planifica esta tarea para realizar mejoras sobre los mismos. Además se añaden botones en forma de flechas (hotspot) para que el usuario se pueda mover de un panorama a otro y pueda hacer una visita virtual del espacio en el que se encuentre o que desee observar.

   Este objetivo fue asignado a Víctor Ortiz y Ricardo Pragnell.

3. **Ampliación de funcionalidad: botones leyenda y SOS**

   Se plantean mejoras sobre la funcionalidad de la aplicación, dando como resultado los nuevos botones: leyenda y SOS. El primero se incluye en el mapa e indica la leyenda de los iconos que aparecen en el mismo, mientras que el segundo localiza al usuario y le dirige a la salida de emergencia más cercana.

   Este objetivo fue asignado a Miguel Gutiérrez y Víctor Ortiz.

4. **Realización de pruebas y corrección de errores en la aplicación móvil**

   Esta tarea se planifica para realizar varias pruebas sobre la aplicación y sobre la herramienta Map Editor, y así hacer las correcciones necesarias de los errores que se encuentran en ambas partes.

   Este objetivo fue asignado a Miguel Gutiérrez y Víctor Ortiz.

5. **Realización de pruebas y corrección de errores en el sistema de reconstrucción**

   Se plantea mejorar lo máximo posible los mapas generados, así como mejorar detalles técnicos y visuales del sistema de reconstrucción. Se realizan una serie de pruebas exhautivas para comprobar que el sistema de reconstrucción funciona correctamente.

   Este objetivo fue asignado a Alejandro Peñalver y Ricardo Pragnell.

3.4. Control y seguimiento del proyecto

En esta sección se describen todos los planes que sirven de apoyo al desarrollo del proyecto, que ayudan a su realización y su control para que se puedan cumplir con los objetivos de cada una de las iteraciones. Entre ellos se encuentran el plan de informes, un gestor de versiones tanto para la documentación como para la implementación del proyecto, así como un plan de riesgos que el equipo ha seguido para no encontrarse con imprevistos y ser capaces de realizar las entregas en su fecha estimada.

Plan de requisitos

Es importante para este proyecto ver como los requisitos influyen en el desarrollo. Al tratarse de un tema desconocido por los integrantes del equipo, se propusieron una serie de requisitos al principio del proyecto que difieren de los requisitos finales. El equipo de ANVM ya contaba con esto, y por tanto decidió dedicar dos iteraciones del proyecto a investigar, para poder fijar los requisitos lo antes posibles.

Todo esto hace que a pesar de no llevar ningún control exhaustivo sobre los requisitos del proyecto, sí que se tuvo muy en cuenta que podrían cambiar durante el desarrollo del proyecto, bien porque no se hubiera investigado lo suficiente, o porque directamente algun requisito fuera irrealizable con respecto a la fecha de finalización del proyecto. Se pueden consultar los requisitos de este proyecto en la sección de Requisitos.

Plan de control de la planificación

Para cada iteración se preparaba una lista con los objetivos que se querían cumplir según la planificación global y el tiempo disponible por parte de cada integrante del equipo. La mayoría de los objetivos que se pusieron para las iteraciones se fueron cumpliendo, así que por lo general no hubo que tomar medidas ante la falta de realización de objetivos.

No obstante para aquellas ocasiones en las que fue necesario se seguía un plan de acción bien sencillo dado que este equipo está integrado tan sólo por 4 desarrolladores. Estas eran las dos pautas a seguir a la hora de asignar objetivos que no hubieran sido cumplidos a tiempo:

- Si el objetivo no había sido cumplido por falta de tiempo dentro del periodo de realización, se volvía a asignar la tarea a la/las mismas personas como tarea extra para la iteración siguiente.
- Si el objetivo no había sido cumplido porque la/las personas a las que se les había asignado no habían sido capaces o fuera una tarea demasiado grande para un grupo demasiado reducido, entonces se le daba prioridad absoluta en la siguiente iteración y se añadían los desarrolladores del equipo que fueran necesarios a dicha tarea.

Estas dos pautas dieron buen resultado, pues si no se cumplía la planificación en un cierto momento, se recuperaba en seguida en la siguiente iteración y la planificación no sufría mayores problemas.

Plan de evaluación y control de calidad

El plan de evaluación de este proyecto es muy simple y consiste en la realización de pruebas contínuas sobre el desarrollo. Dichas pruebas se realizaron fundamentalmente:

- Al realizar cada uno de los objetivos de cada una de las iteraciones. Una vez terminado un objetivo, que cumplía con una parte de la funcionalidad de este proyecto, se realizaban una serie de pruebas para comprobar que dicho objetivo había sido cumplido satisfactoriamente. Estas pruebas no eran demasiado exhaustivas puesto que sino hubieran retrasado enormemente la realización del proyecto.
- Al finalizar el primer prototipo. Dado que esta fue la primera de las versiones más o menos estables del sistema, se planifica realizar una serie de pruebas más exhaustivas, que comprueben que todos los elementos implementados durante las iteraciones anteriores funcionaran correctamente y en armonía unos con otros.
- Al finalizar el segundo prototipo. Puesto que a partir de este punto no se iba a implementar más funcionalidad, se planificaron pruebas similares a las del primer prototipo, aunque aún más exhaustivas intentando dejar el proyecto totalmente funcional. A este tipo de pruebas se las dedicaron más tiempo debido a su importancia para el prototipo final (éste mismo pero con todos los errores posibles corregidos) en torno a 3 semanas al final del proyecto.

A pesar de ser un proyecto relativamente corto y ser pocos desarrolladores, se intentó realizar la mayor cantidad de pruebas posibles para que el proyecto fuera sólido. Se intercambiaron los papeles a la hora de realizar las pruebas, es decir, todos los integrantes del equipo intentaron probar todos los elementos del proyecto a pesar de no haberlos implementado ellos mismos, para intentar descubrir más errores.

Así mismo para el caso de la aplicación móvil se instaló la última versión a los familiares de los integrantes del equipo para que aportaran también información de errores que se les pudieran producir.

Plan de informes

Toda la comunicación interna de este proyecto se realiza utilizando Google Docs, un sistema de colaboración en tiempo real en línea. También se realizan pequeños informes sobre lo que cada miembro ha implementado día tras día, gracias a la herramienta TortoiseSVN, sistema de control de versiones.

En cada iteración se establecían unos objetivos y fechas límite que debían respetarse por todos los miembros del equipo, establecidos en diversos documentos creados en Google Docs. Por otro lado los desarrolladores deben presentar informes cada semana o cada dos semanas con los objetivos conseguidos y las dificultades encontradas, para facilitar así la comunicación entre ellos y la solicitud de ayuda en caso de que un miembro hubiera encontrado una gran dificultad a la que no pudiera hacer frente él sólo.

La comunicación externa con los profesores se realizó mediante Dropbox, en donde se almacenaba la información de las progresivas investigaciones, y mediante correos electrónicos con los que se concretaban pequeñas reuniones para mostrar los progresos realizados y consulta de preguntas o sugerencias.

Plan de riesgos

No se siguió ningun plan de riesgos específico para este proyecto. Cada uno de los problemas o riesgos que se detectaban a lo largo del proyecto se iban apuntando en un documento de riesgos en Google Docs como se indica a continuación. Estos riesgos se trataban de resolver lo antes posible si eran riesgos críticos. Por el contrario si el riesgo se consideraba asumible o no de mucha importancia, se dilataba su resolución hasta que alguno de los integrantes del equipo estuviera más liberado de trabajo y puediera darle una solución.

Plan de documentación

Sin duda este plan fue fundamental tanto para tener un seguimiento del proyecto, de lo que se iba investigando y desarrollando, así como escribir la planificación que posteriormente queda reflejada en esta misma sección. El plan que se siguió para realizar la documentación del proyecto es bien sencillo.

En primer lugar durante todo el transcurso del proyecto se utilizaron varios documentos en el sistema Google Docs en los que cada integrante del grupo iba actualizando información según iba trabajando. Más adelante, en el transcurso del proyecto, se procedió a pasar dichos documentos con toda la información sobre el proyecto a este documento, redactando de mejor manera la información contenida en los Google Docs y dándole la estructura que tiene esta memoria. A continuación se resumen los documentos creados y cuál era su cometido.

- **Documento de investigación**

  Documento clave durante la primera fase del proyecto, en el que se iban escribiendo todos los hallazgos en cada uno de los campos investigados. El formato que se seguía en este documento era sencillo, constaba de un título del tema que se había investigado, y pequeño párrafo explicando los resultados hallados, y una serie de links donde poder encontrar información complementaria al tema que se había investigado. De esta manera fue después bastante sencillo documentar cada uno de los hallazgos de la investigación.

- **Documento de riesgos**

  Cada vez que un integrante se topaba con un nuevo riesgo, este era actualizado en la tabla de riesgos, junto con una pequeña descripción del mismo. Si el riesgo fue solucionado en algún momento se comentaba su solución, y de no ser así se dejaba como riesgo no resuelto.

- **Documento de requisitos**

  Documento muy sencillo que contenía una lista de los requisitos del sistema junto con una descripción de cada uno de ellos. A pesar de ser un documento muy corto, fue actualizado una gran cantidad de veces sobre todo durante el periodo de investigación, y fue muy útil durante el desarrollo para mantener claros los objetivos y la funcionalidad que había que desarrollar.

- **Documento de casos de uso**

  Una vez pasada la fase de investigación donde los requisitos quedaron más o menos claros, se creó un documento donde se describían brevemente cada uno de ellos, para fijar conceptualmente dichos casos de uso. Este documento trataba de fijar la funcionalidad que se reflejaba en el documento de requisitos, y facilitar el diseño del sistema.

- **Documento de arquitectura**

  El objetivo de este documento es claro: reflejar todos los cambios que se produjeran en el diseño o implementación del sistema. Se describían convenios establecidos para ciertos temas, de manera que todo el equipo siguiera las mismas pautas dentro del proyecto, así como una pequeña lista con los elementos clave de la arquitectura del sistema según se iban diseñando e implementando.

- **Documento de planificación**

  Fue quizás el documento más actualizado donde se colgaba cada semana la planificación de los objetivos a realizar y quién debía de realizarlos. Según se iban realizando se iban tachando las tareas de la lista, y fue muy útil dado que de esta manera todo el mundo sabía en lo que estaba trabajando cada uno y a quién se tenía que preguntar cosas según qué tema. Fue de gran utilidad también para la realización de esta misma sección de planificación.

- **Documento de documentación**

  Valga la redundancia, el objetivo de este documento era similar al anterior, organizar los objetivos a realizar por cada integrante del grupo, pero en este caso para la documentación. Se utilizaba una lista con objetivos semanales de la misma manera que para la planificación del desarrollo.

Capítulo 6

Proceso de reconstrucción

El objetivo de este capítulo de la memoria es dejar totalmente claro cuál es el procedimiento a seguir para llevar a cabo una reconstrucción con el sistema ANVM. De la misma manera se pretende explicar cada uno de los procesos que se llevan a cabo durante la reconstrucción, así como las configuraciones necesarias para cada uno de dichos procesos. Se explicarán también los algoritmos utilizados. Si tras leer esta sección el lector tuviera alguna duda de cómo se implementan dichos algoritmos puede recurrir a la sección de Arquitectura.

1. Conceptos teóricos

En esta sección se explicarán todos los conceptos necesarios para comprender cómo funciona el sistema de reconstrucción y captura de datos. Se detallarán los algoritmos y métodos empleados para obtener la información mediante el robot de captura de datos, así como todos los aspectos teóricos que conforman este sistema.

1.1. Captura de datos de odometría

Como se ha comentado anteriormente, la odometría corresponde a la información sobre la posición del robot. Esta información es necesaria para llevar a cabo la reconstrucción, puesto que necesitamos ir conformando las imágenes tomadas junto con la posición desde la que se tomaron. Se van a explicar a continuación los elementos principales de este robot, y a través de ellos se comprenderá cómo se lleva a cabo el cálculo de la posición.

Encoders

El elemento principal del robot para calcular esta posición son los llamados encoders. Se trata de un elemento del tamaño de la tapa de un bolígrafo aproximadamente. Dispone de un cilindro con varios puntos de conexión eléctricos que al rotar envían pulsos. En la figura 36 se puede ver una imagen de este dispositivo (izquierda). El cilindro plateado de la parte superior es el que va girando, tanto hacia un lado como hacia otro. Cada cierto grado de giro se produce una conexión que activa una de las salidas del encoder y emite un pulso. Si hacemos girar este cilindro repetidamente se producirán una serie de pulsos que nos permitirán determinar cuantas vueltas ha dado el encoder.

En la imagen de la derecha de la figura 36 se puede ver rodeado con un círculo naranja el lugar donde está situado cada encoder en cada rueda. La rueda del carro (la grande y gris que está en contacto con el suelo) rota al avanzar con el robot, y hace girar a la rueda negra que a su vez hace girar a las ruedas engranadas grises más pequeñas. Estas ruedas son las que llevan acoplado el encoder y por tanto lo hacen moverse.

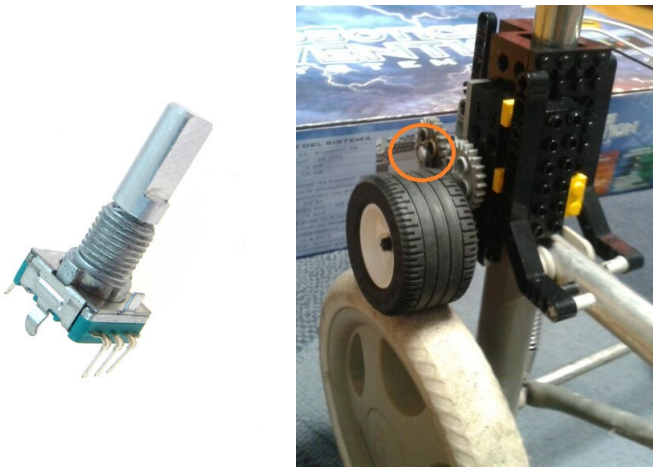

Figura 36: Encoder (izquierda). Encoder montado en el robot (derecha).

El robot dispone de dos encoders, uno en cada rueda, que giran a la vez que la propia rueda del carro. De esta manera si determinamos que una vuelta de la rueda del carro corresponde con un cierto número de ticks del encoder, no tenemos más que contar el número de ticks totales para determinar la distancia exacta que nos hemos movido. Por ejemplo, supongamos que una vuelta del carro corresponde con 20 ticks del encoder. Recorremos una cierta distancia en línea recta con el carro de manera que recogemos 200 ticks en cada encoder, por tanto la rueda del carro ha dado 200 / 20 = 10 vueltas. Conociendo el radio de la rueda (pongamos 7 cm) podemos hallar la longitud de la circunferencia de la rueda con la conocida fórmula 2 * Π * radio y obtenemos una longitud de 43.96 cm. No tenemos más que multiplicar el número de vueltas (10) que ha dado la rueda por su longitud y obtenemos como resultado 439.6 cm. Este supuesto es para el caso de andar en línea recta, y por tanto ambos encoders producen el mismo número de pulsos (ambas ruedas se mueven la misma distancia). Sin embargo esta aproximación no nos sirve si realizamos giros con el carro, puesto que cada rueda se mueve un número de vueltas diferente.

Giros con el robot

Para tener en cuenta que el robot puede realizar giros necesitamos generalizar la fórmula que acabamos de comentar. La cuestión está en que cada rueda se puede mover libremente y no tienen por qué moverse lo mismo. Por ejemplo, si vamos avanzando con el carro y realizamos un giro a la derecha, la rueda izquierda se moverá hacia adelante mientras que la rueda derecha quedará inmóvil o se moverá menos. Por tanto el encoder de la rueda izquierda contará un número de ticks mayor que el de la rueda derecha. Teniendo en cuenta esa diferencia de ticks entre un encoder y otro podemos determinar el ángulo exacto de giro así como la distancia recorrida. A continuación podemos ver las ecuaciones que nos permiten calcular la posición actual del robot:

$$\Phi = (K_{right} * encticks_{right} - K_{left} * encticks_{left})/wheelsDistance \quad (1)$$

$$As = 0{,}5 * (K_{right} * encticks_{right} + K_{left} * encticks_{left}) \quad (2)$$

$$x = cos(\Phi) * As \quad (3)$$

$$y = sin(\Phi) * As \quad (4)$$

Estas ecuaciones calculan cuánto ha avanzado el robot en una iteración, desde que se captura un cierto número de ticks de los encoders hasta que se realiza la siguiente lectura. En la ecuación (1) se calcula el ángulo del giro (Φ) que ha dado el robot desde la anterior lectura. Las constantes K_{right} y K_{left} representan la distancia que se mueve la rueda izquierda y derecha (respectivamente) por cada tick del encoder y viene expresada en metros. Las variables $encticks_{right}$ y $encticks_{left}$ representan el número de ticks que se han producido en cada encoder desde la anterior lectura. Por tanto la multiplicación de cada variable K por su encticks correspondiente nos proporciona la distancia que se ha avanzado con cada rueda. Finalmente $wheelsDistance$ se refiere a la distancia entre ambas ruedas expresada en metros también. Cabe destacar que si hemos avanzado en línea recta la distancia recorrida por ambos encoders sería la misma, y por tanto el numerando de (1) sería 0, lo que nos dejaría un ángulo de giro Φ=0.

En la ecuación (2) calculamos la distancia total avanzada. De manera similar a la ecuación anterior, si hemos avanzado en línea recta la distancia recorrida por ambas ruedas es la misma, por lo que tendríamos una ecuación equivalente a: $As = 0{,}5 * (2 * distancia_{rueda})$ de manera que al multiplicar por 0.5 obtenemos la distancia recorrida por una sola rueda. Por último en (3) y (4) calculamos el incremento de x y de y sobre la posición de la iteración anterior. No tenemos más que sumar estos incrementos a dicha posición anterior y obtendremos la nueva posición del robot.

Estas cuentas se implementan en el método computeFromEncoders de la librería MRPT, sin embargo el cálculo que se realiza no es exactamente el mismo. Se calcula el incremento de la posición exactamente como acabamos de explicar, en cambio después de esto se le aplica un "modelo de movimiento". Se trata de un modelo probabilístico que trata de corregir pequeños errores y mejorar la precisión en el cálculo de la posición.

Placa arduino

La misión de este microprocesador es recoger los pulsos emitidos por cada encoder e ir incrementando un contador por cada uno. Disponemos por tanto de un programa cargado en el arduino que nos guarda estos dos contadores y que cada cierto tiempo los va transmitiendo por el USB. Estos valores de los contadores serán leidos por el programa de reconstrucción que se ejecuta en el ordenador para realizar las cuentas de la posición que acabamos de explicar.

1.2. Captura de datos de la imagen

Como ya sabemos, para construir el mapa de la superficie necesitamos información sobre la posición del robot, y sobre la imagen que se está capturando en ese momento. El dispositivo del robot encargado de realizar la captura de la imagen es el Kinect. Como ya hemos visto en la sección de investigación este dispositivo es capaz de capturar nubes de puntos 3D que describen el espacio que se está visualizando en el momento. Dado que lo que nosotros queremos es conseguir simular un láser en 2D, deberemos transformar esa nube de puntos de alguna forma.

Observación 3D a observación 2D

En este apartado vamos a explicar cómo se realiza el paso de nube de puntos 3D a 2D. Imaginemos que nuestro dispositivo Kinect ha recogido una nube de puntos similar a la imagen de la figura 37. Podemos dividir esta nube en una serie infinita de planos horizontales, donde cada punto de la nube pertenece a un plano. Un escáner láser 2D apuntando al mismo sitio, habría recogido los puntos de la nube que pertenecieran a un solo plano horizontal de los infinitos existentes.

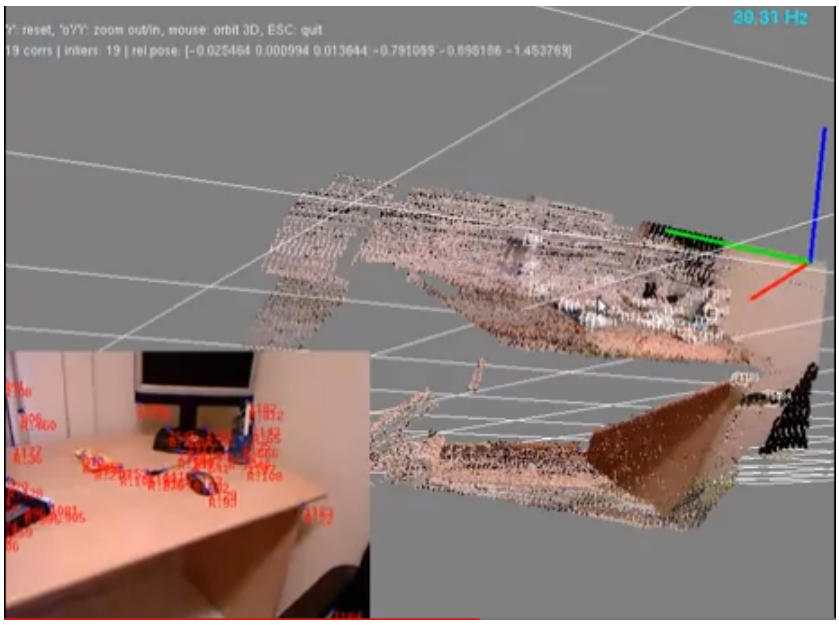

Figura 37: Vídeo de la reconstrucción (izquierda). Nube de puntos (derecha).

Siguiendo esta idea, lo único que tenemos que hacer para conseguir simular un escáner láser en 2D es trazar un plano horizontal sobre la nube de puntos en 3D y quedarnos con los puntos en 2D que intersecccionen con ese plano. Sería similar a "cortar" la escena en 3D mediante un plano horizontal y obtener una vista cenital de dicha escena. Concatenando estas vistas cenitales obtendremos finalmente el mapa 2D que deseábamos construir.

Esta idea es muy sencilla, sin embargo la técnica que se utiliza no es exactamente ésta. Según acabamos de explicar, podríamos coger cualquier plano horizontal sin embargo

el resultado no sería el mismo, dependiendo del plano que escogiéramos obtendríamos unos puntos u otros. La técnica que se sigue (ya que se dispone del conjunto de puntos en 3D) consiste en proyectar los puntos estos puntos en 3D al plano horizontal por el que hemos cortado. Aquellas secciones donde caigan un mayor número de puntos, representarán elementos más importantes de la escena, para nuestro caso, paredes y obstáculos.

1.3. Algoritmo de reconstrucción

Existen varios algoritmos capaces de construir un mapa a través de información de odometría e imagen. En este proyecto utilizamos el algoritmo "rbpf slam". Las siglas corresponden a Rao-Blackwellised Particle Filtering, un tipo de algoritmo basado en el filtrado de partículas. La librería MRPT dispone de una implementación de este algoritmo de forma offline, es decir, trabaja con información previamente guardada en un fichero y a partir de ahí crea el mapa, no trabaja en tiempo real. Actualmente en el proyecto se utiliza estar versión del algoritmo, introduciéndole como fichero los datos capturados anteriormente robot (Kinect + Arduino), sin embargo no se descarta en un futuro utilizar otra versión que permita construir el mapa en tiempo real. A continuación vamos a comentar brevemente cómo funciona este algoritmo. Si el lector tiene alguna duda puede consultar la implementación de este algoritmo en la página web de la librería [18].

RBPF SLAM (Rao-Blackwellised Particle Filtering)

Se trata de un algoritmo basado en el filtro de partículas con algunas optimizaciones para conseguir mejores tiempos de ejecución. En primer lugar explicaremos en qué consiste el filtro de partículas. Se trata de un algoritmo basado en redes bayesianas. Una red bayesiana es un grafo acíclico dirigido que representa un modelo probabilístico en el que cada variable (nodos del grafo) dependen de otras. En la figura 38 se puede ver una red bayesiana con tres variables: lluvia, rociador, y hierba húmeda. La variable Lluvia condiciona al rociador y a la hierba húmeda y la variable rociador condiciona a la hierba húmeda. Por tanto las probabilidades del rociador y de la hierba húmeda están condicionadas por la lluvia.

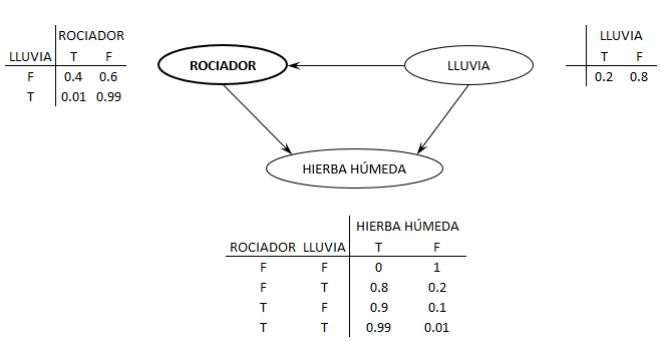

Figura 38: Red Bayesiana simple

Los filtros de partículas se utilizan para estimar redes bayesianas. En cada iteración se tratan de filtrar las partículas de manera que sobrevivan las que más probabilidad tengan de ser correctas con respecto a una función de probabilidad definida. En la figura 39 podemos ver un ejemplo de cómo funciona el filtro de partículas. Al inicio de la iteración disponemos de un conjunto de particulas (círculos amarillos de arriba), donde las más probables son las más grandes como indica la función de probalididad que aparece justo encima. Escogemos una muestra (verdes de arriba) de ese conjunto de partículas (teniendo más en cuenta las más probables) y propagamos estas partículas según un modelo de transición definido obteniendo las partículas verdes de abajo. Después ponderamos esa muestra propagada con los datos disponibles para esta iteración y tomamos una muestra que corresponde con los círculos amarillos de abajo y es el resultado del filtrado.

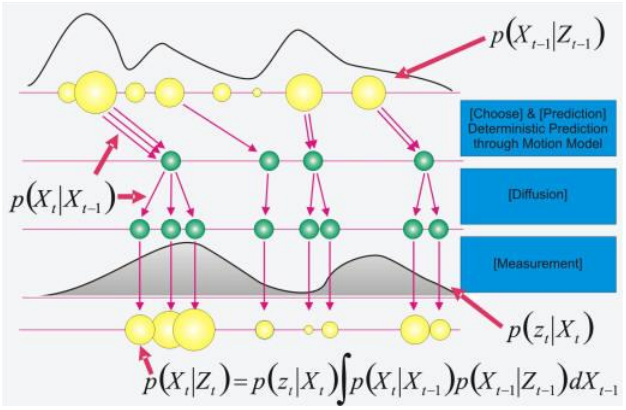

Figura 39: Filtro de partículas

Aplicado al slam, las partículas se corresponden con los diferentes mapas que se han reconstruido hasta el momento. Todas las partículas tratan de representar el mismo mapa real, sin embargo debido a la evolución de los ciclos de filtrado, cada partícula contiene un mapa diferente que se corresponderá más o menos con la realidad. El objetivo es conseguir el mapa que mejor represente la realidad. El algoritmo rbpf realiza un filtrado de partículas como el que acabamos de explicar, pero trata de optimizarlo reduciendo el número de partículas a tener en cuenta.

Dado que cada partícula guarda toda la información referente a un mapa completo este ahorro puede ser muy significativo. La manera de reducir estas partículas es quedarse con aquellas cuya posición sea la más cercana a la posición capturada desde el robot (información de odometría). De esta manera, estamos premiando las partículas cuya trayectoria en el mapa les haya llevado a una posición más cercana a la que el robot dice estar para esa iteración. Por tanto no estamos perdiendo información relevante, dado que las partículas cuyas posiciones son las más alejadas a la posición actual del robot son las eliminadas.

Cabe destacar por tanto que es muy importante que la información sobre la posición del robot (odometría) sea lo más precisa posible. Datos de odometría precisos llevan a mejores reconstrucciones puesto que el algoritmo rbpf filtrará mejor las partículas y el mapa generado será más similar a la realidad.

2. Cómo llevar a cabo una reconstrucción

En primer lugar vamos a describir los elementos necesarios para poner en marcha el sistema:

- Disponer del robot fabricado por los integrantes de este proyecto.
- Tener cargado el programa de captura de datos *encoder.ino* en la placa Arduino Uno del robot que acabamos de mencionar.
- Disponer de un ordenador portátil al que se conectará el robot y donde se pueda ejecutar el programa de reconstrucción.

Una vez que tenemos todos estos elementos listos podemos comenzar con la captura de datos.

2.1. Captura de datos

Arrancamos la aplicación *ANVM_VirtualMapper_GUI.exe* que nos muestra la interfaz principal del sistema de reconstrucción (ver 40). Una vez en esta interfaz debemos rellenar los campos de la sección de captura, en la sección superior. Le daremos un nombre al proyecto, y seleccionaremos un directorio de trabajo donde guardar los datos de la captura y reconstrucción.

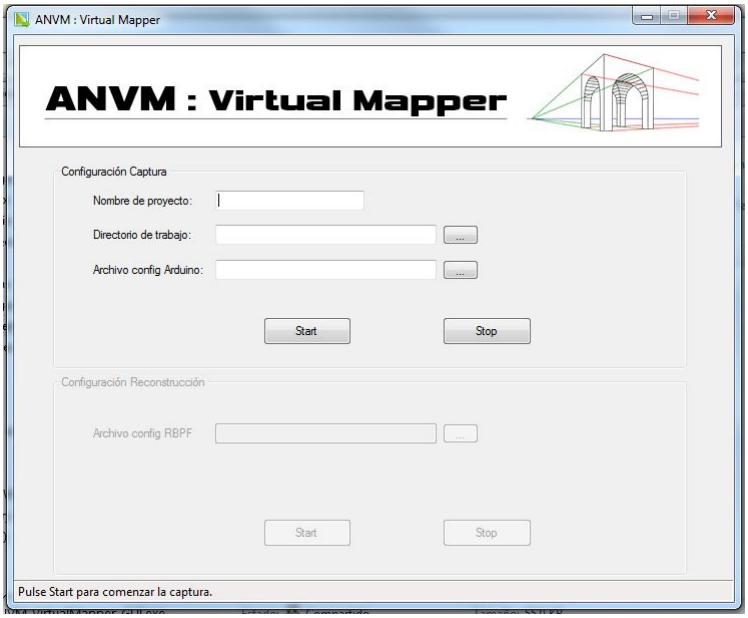

Figura 40: Interfaz inicial

Seleccionaremos también el fichero de configuración del robot. consiste en un fichero de texto que debe ser similar al que se muestra en la figura 41. Gracias a este fichero se pueden configurar los siguientes parámetros:

- Intervalo de tiempo entre lecturas del arduino expresado en milisegundos. Hay que tener en cuenta que un elevado tiempo puede hacernos perder información valiosa.

- Relación metros/ticks del encoder de la rueda izquierda y derecha. Hay un parámetro diferente para cada rueda puesto que una puede girar un poco más rapido que la otra y viceversa.

- Distancia entre las ruedas del robot expresada en metros.

```
////////////////////////////////////////////////////////////
////////////////// CONFIG ARDUINO FILE /////////////////////
////////////////////////////////////////////////////////////

Time interval for reading encoder ticks (ms) = 50
Left encoder ratio (meters/ticks) = 0.002605789473684
Right encoder ratio (meters/ticks) = 0.00253667068757
Distance between both wheels (meters) = 0.38
```

Figura 41: Archivo configuración arduino

Una vez hemos rellenados estos campos pulsaremos el botón *Start* y comenzaremos con la reconstrucción. Nada más pulsar este botón nos aparecerá una nueva ventana en la que podremos ver el espacio que estamos reconstruyendo. Podemos ver una captura de esta ventana en la figura 42. En esta pantalla se nos muestran también datos sobre el espacio que estamos reconstruyendo. En la parte inferior izquierda de la imagen se observa un video de lo que se está capturando en el momento. Sobre este video salen una serie de datos en rojo que corresponde con los puntos característicos de la escena. En la parte de arriba aparece la observación o nube de puntos de la reconstrucción con los mismos datos en color blanco.

Para avanzar en la reconstrucción no tendremos más que llevar el carro del robot por el espacio interior que deseamos reconstruir. Durante la reconstrucción tenemos que tener en cuenta que el Kinect dispone de un alcance limitado de unos 6 metros. Por tanto tendremos que acercar el Kinect a esta distancia de todos los objetos que queramos que aparezcan en la reconstrucción. Continuaremos llevando el carro por todo el espacio interior y cuando hayamos acabado pulsaremos el botón *Stop*.

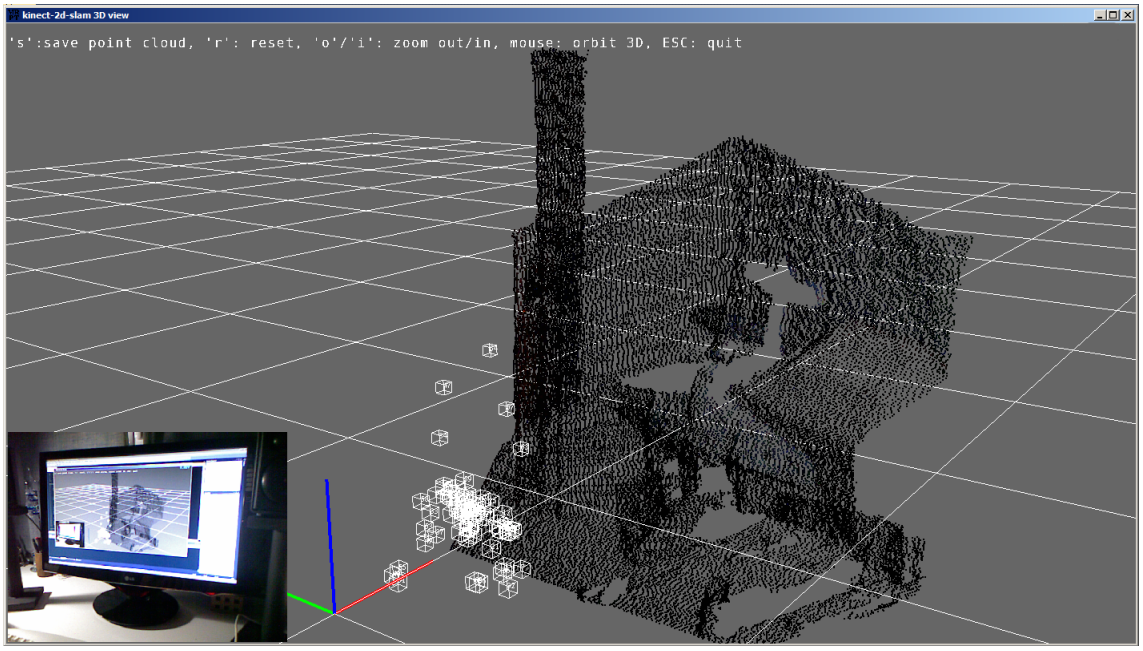

Figura 42: Ventana durante la reconstrucción

Una vez finalizada la ejecución se crearán dos carpetas en el directorio de trabajo que hemos introducido: la carpeta *dataset*, y la carpeta *log*. En la carpeta dataset se creará un fichero con el nombre del proyecto y de extensión *.rawlog*. En este fichero se guardan todos los datos que acabamos de capturar y que será la entrada del generador de mapas. En la carpeta log se habrán creado tres ficheros correspondientes a los ficheros de log de los tres hilos de ejecución del sistema.

2.2. Generación del mapa

Una vez hemos recorrido el espacio interior con el robot y hemos capturado todos los datos necesarios, procedemos a la generación del mapa. Se nos cerrará la ventana de la figura 42 y volveremos a la pantalla inicial que se muestra en la figura 40. Esta vez aparecerá activada la sección inferior de la interfaz, donde se nos solicitan los datos necesarios para construir el mapa. En este caso el único dato que hay que introducir es el fichero de configuración para el algoritmo rbpf slam. Este fichero permite configurar todas las opciones del algoritmo, entre ellas cabe destacar algunas:

- Cotas para ver con cuanta frecuencia se insertan observaciones en el mapa que se va generando. Nos permite calibrar cómo de minucioso es el algoritmo.
- Tipo de filtro de partículas a utilizar. Para este proyecto se utiliza el pfAuxiliaryPFOptimal. Se puede consultar los detalles de esta aproximación en la página web de la librería MRPT.
- Dependiendo del tipo de algoritmo que se desee utilizar para el filtro de partículas se nos permite configurar unos parámetros u otros referentes a dicho algoritmo.

En la página web de la librería MRPT se ofrecen diferentes ficheros de configuración de ejemplo. Para este proyecto se utiliza un fichero modificado y adaptado por los desa-

rrolladores del proyecto con el que sea han realizado las pruebas que se describen en la sección de Plan de pruebas. Finalmente tras haber seleccionado este fichero de configuración pulsaremos el botón Start de la sección inferior. El sistema se encargará de buscar el fichero .rawlog generado anteriormente en el proceso de captura de datos, y llamará al algoritmo de recnostrucción. Se mostrará una barra de carga en la parte inferior derecha de la interfaz principal y cuando finalice la generación del mapa se mostrará automáticamente el resultado. El mapa generado será similar al de la figura 43, y se habrá guardado en la carpeta rbpf-slam/maps dentro del directorio de trabajo del proyecto.

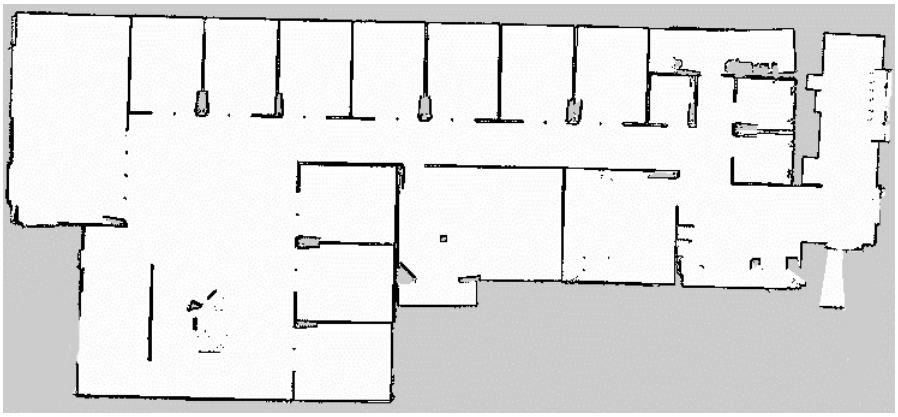

Figura 43: Mapa construido mediante rbpf slam

Capítulo 7

Arquitectura

Este capítulo de la memoria proporciona una visión general de la arquitectura del sistema de reconstrucción, utilizando para ello distintas vistas que describen diferentes aspectos del sistema. El objetivo es dejar claras todas las decisiones de diseño que se han tomado para esta arquitectura, y por qué se han tomado. De la misma manera también se pretende mostrar el funcionamiento interno del sistema con un nivel de detalle suficiente que permita la comprensión del mismo.

Como se verá más adelante en esta sección, se divide la arquitectura del sistema de reconstrucción en dos bloques claramente diferenciados:

1. **Arquitectura del robot**

   Corresponde con la parte de la captura de datos necesarios para llevar a cabo la reconstrucción. Se centra en el diseño y construcción del robot que permite realizar la captura, así como el software que hace funcionar dicho robot y que permite enviar los datos capturados al siguiente módulo.

2. **Programa de reconstrucción**

   Se trata del software encargado de procesar los datos capturados y llevar a cabo la reconstrucción. Se encarga también de llevar una comunicación con el robot que captura los datos para poder guardarlos y procesarlos posteriormente. Dispone de una interfaz que sirve de comunicación con el usuario que está realizando la reconstrucción.

Todas las palabras, tecnicismos, siglas y expresiones que necesiten de una mayor explicación están recogidas en el Glosario.

1. Objetivos y restricciones de la arquitectura

En esta subsección se describen brevemente los objetivos que se persiguen con esta arquitectura, así como las restricciones que presenta y cómo influyen éstas en el diseño del sistema.

1.1. Objetivos

Tal y como se explica en la sección de Investigación, los esbozos de la primera arquitectura del sistema fueron pensados de distinta manera a como es la arquitectura en la actualidad, puesto que se pretendían resolver problemas diferentes. Tras el periodo de investigación la arquitectura se estabilizó, pero a pesar de todo, los objetivos de la misma han permanecido inmutables desde el principio y son los siguientes:

- **Modularización**

  Esta arquitectura es ya de hecho un módulo de una arquitectura mayor que engloba los dos proyectos de ANVM. Además, para esta arquitectura, la modularización es algo totalmente necesario puesto que se pretende integrar diferentes elementos dentro del mismo sistema. Existe por tanto diferenciación de módulos entre el hardware del sistema (el robot) así como en los diferentes procesos que se han de llevar a cabo concurrentemente y que se explicarán más adelante. Con todo ello se consigue una mayor eficiencia en el desarrollo, siendo realizadas varias tareas en paralelo por diferentes integrantes del equipo. Además, facilita realización de pruebas sobre el sistema dado que cada módulo se puede probar por separado.

- **Escalabilidad**

  Desde un principio este proyecto se pensó como un comienzo o primera aproximación para resolver un problema mayor que es la virtualización en 3D. Puesto que esto se investigó y se optó por comenzar a desarrollar un sistema que reconstruyera en 2D, cada uno de los elementos de la arquitectura se han pensado para poder ser ampliables en un futuro. Por tanto la escalabilidad del sistema es uno de los objetivos primordiales, ya que permitirá ampliar la arquitectura para resolver problemas mayores, partiendo de la arqutiectura base del proyecto.

1.2. Restricciones

Existen varios tipos de restricciones sobre la arquitectura de este proyecto. Las dividiremos en dos categorías y las explicaremos brevemente:

- **Restricciones de diseño e implementación**

  Corresponden con todas aquellas restricciones que vienen impuestas por la estrategia de implementación que se ha seguido, así como de las herramientas y librerías por las que se han optado para implementar el sistema, y son las siguientes:

  - La arquitectura se debe comunicar correctamente con la librería MRPT puesto que el sistema de reconstrucción está montado sobre ella. Esto hace que los datos capturados deban ser guardados en un formato específico para que esta librería los pueda interpretar posteriormente.
  - Debido a que es necesaria la ejecución de varios procesos en paralelo, la arquitectura debe permitir la ejecución de varios hilos a la vez, y garantizar la coordinación y comunicación entre los distintos hilos.

- **Restricciones de recursos**

  Estas restricciones tienen que ver tanto con los recursos humanos disponibles, es decir, el equipo de desarrolladores que es limitado, así como los recursos hardware que este equipo puede adquirir para el proyecto.

  - Se dispone de cuatro miembros para desarrollar esta arquitectura durante un año teniendo en cuenta los periodos de exámenes para cada uno de ellos. Esto

condiciona el volumen de la arquitectura, ya que se debe ajustar al tiempo y personal disponible.

- Existe una importante restricción sobre los recursos hardware del sistema, y es que no se dispone de presupuesto para adquirir componentes de todo tipo, y por tanto la arquitectura hardware se deberá ceñir a los elementos de los que dispone el equipo de desarrollo.

2. Arquitectura del robot

Durante esta sección explicaremos con detalle cómo está diseñado el robot de reconstrucción, y cómo ha sido el proceso de construcción. Se detallarán los elementos utilizados, así como el cometido de cada uno.

2.1. Construcción Plataforma Móvil:

Para realizar una buena captura de datos es necesario desplazar los sensores de forma estable por el entorno y además almacenar el recorrido realizado. Para realizar este cometido diseñamos una plataforma móvil a partir de un carro de la compra y componentes LEGO ®, ambos económicos y fáciles de conseguir.

Para obtener los datos posicionales se emplean dos encoders rotacionales, uno para cada rueda. El modelo de los encoders usados es el EC11E15204A3, de la marca japonesa Alps [1]. Con una precisión de 15 pulsos por rotación, ofrecen una sensibilidad más que suficiente para nuestras necesidades. En la figura 44 podemos ver dicho encoder y su salida digital, dependiendo del sentido de rotación.

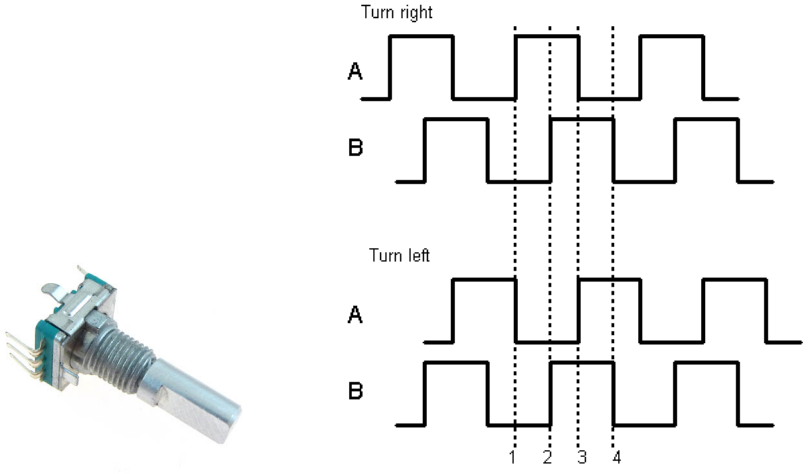

Figura 44: De izquierda a derecha, un encoder de rotación y su salida digital

Uno de los desafíos durante el diseño fue acoplar los encoders a las ruedas del carrito. Para hacer una primera aproximación empleamos un software de diseño específico

para LEGO: LegoCAD [3]. En las figuras 45 y 46 se muestran capturas de una etapa intermedia del diseño del mecanismo de sujeción y acoplo.

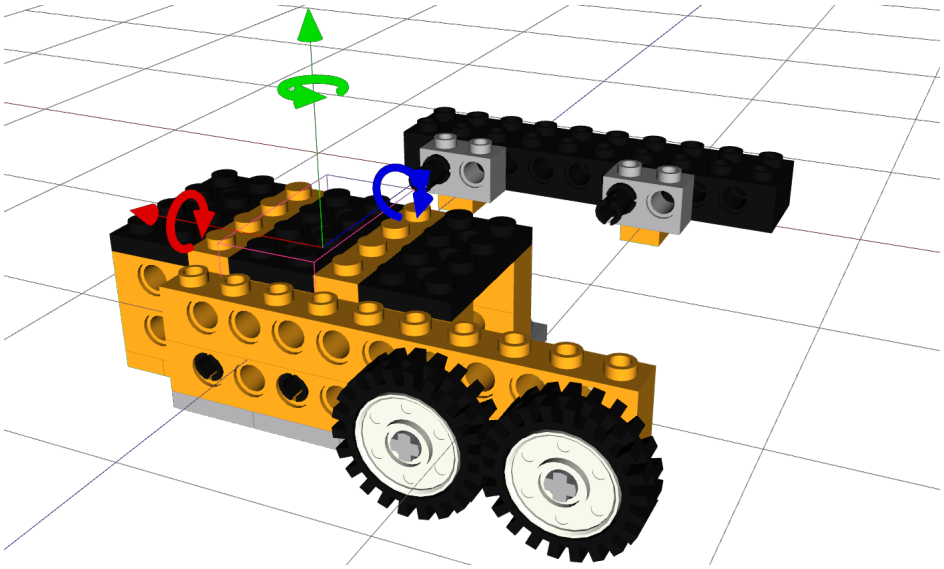

Figura 45: Vista superior del anclaje

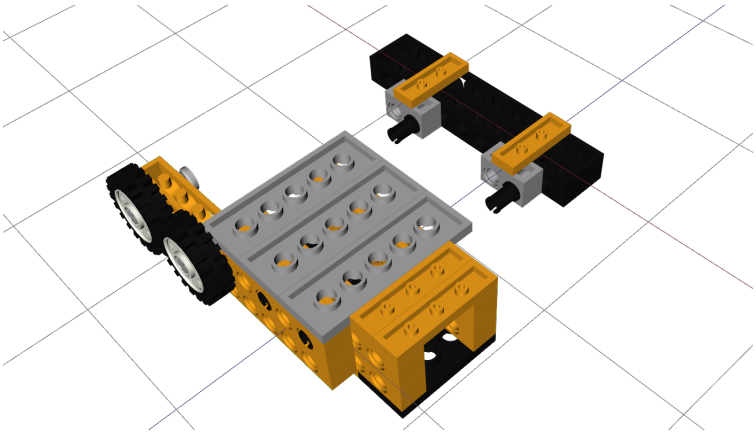

Figura 46: Vista inferior del anclaje

El sistema de sujección se diseñó teniendo en cuenta varias propiedades fundamentales, entre ellas la fácil integración en otras plataformas móviles y la capacidad de regular la precisión de los encoders según distintos tamaños de ruedas (necesario ya que dependiendo del diámetro de la rueda el encoder hará una cantidad mayor o menor de revoluciones por distancia recorrida). Estos requisitos fueron resueltos de la siguiente manera: para el primero se decidió usar una transmisión directa de rotación por fricción, manteniendo contacto directo entre la rueda de la plataforma móvil y la rueda del sistema de sujección. De esta forma logramos eliminar la necesidad de realizar modificaciones a la plataforma móvil quedando un dispositivo modular que solo requiere un punto de contacto con la rueda de dicha plataforma móvil. Para solventar el segundo requisito incorporamos un sistema de engranajes de ejes paralelos compuesto por dos engranajes cilíndricos de dientes rectos. Empleando este mecanismo con distinto núme-

ro de dientes en cada engranaje resulta una forma muy simple de regular la proporción de revoluciones que llega al encoder.

Una vez obtenido un esquema preliminar, la construcción del robot se llevó a cabo como ilustra la figura 47.

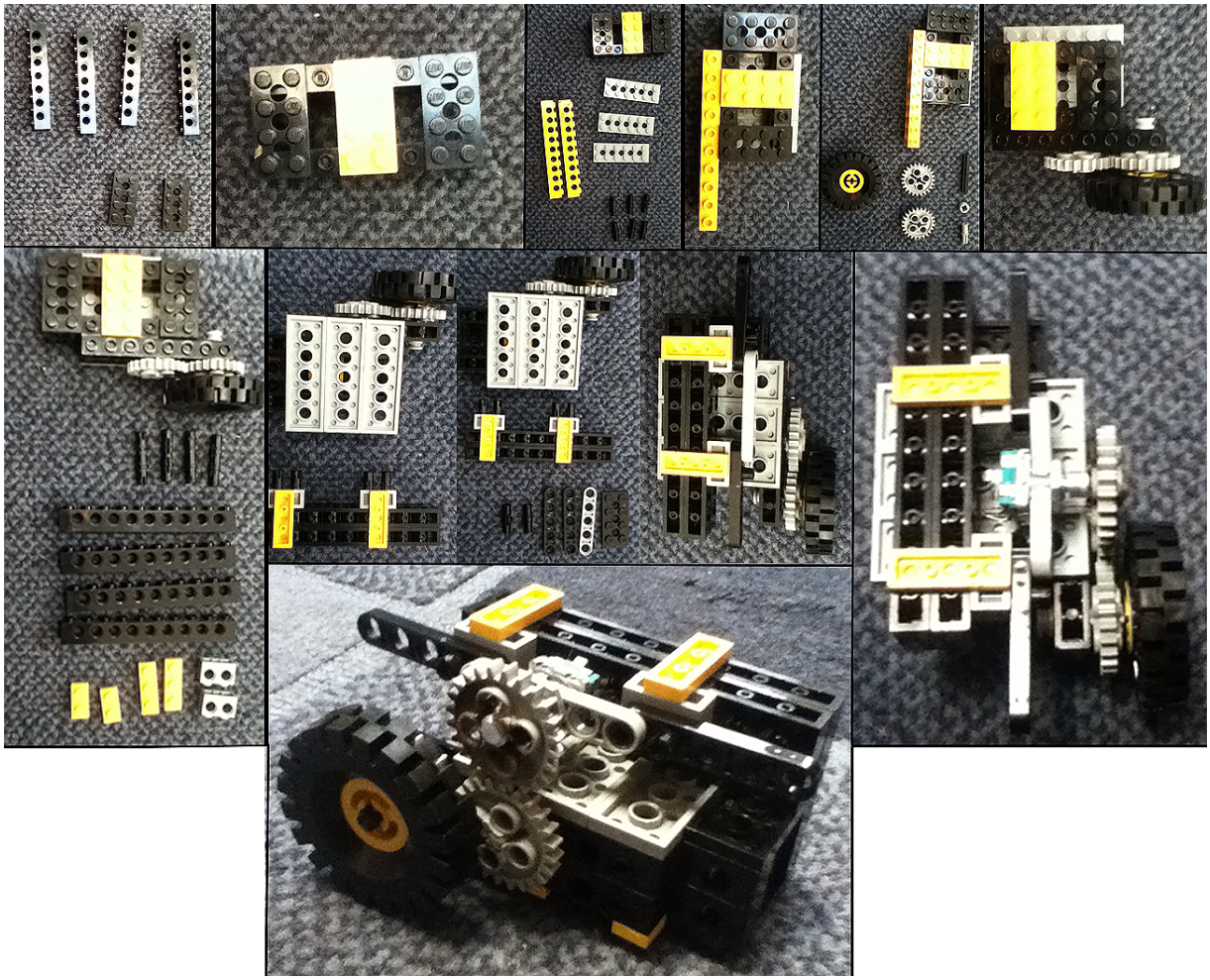

Figura 47: Pasos en la construcción del anclaje para los encoders

Este prototipo resultó ser una buena base desde la cual mejorar, puediendo probar de qué manera influía en la reconstrucción los datos de posicionamiento y los diferentes diámetros de ruedas. Sin embargo tuvimos problemas manteniendo un anclaje firme y constante entre la rueda del encoder y la rueda de la plataforma móvil. Este fallo se produjo principalmente porque la rueda del carrito no era perfectamente circular debido al desgaste y como consecuencia en algunas zonas era más propenso a deslizar y no transmitir la rotación correctamnete al encoder. Identificado el problema, optamos solucionarlo mediante muelles conseguiendo un buen anclaje sin entorpecer el movimiento de las ruedas del carro. La figura 48 muestra el resultado final de dicho anclaje.

Figura 48: Anclaje mediante muelles

En la última iteración del diseño se añadieron piezas LEGO adicionales para reforzar la estructura y restringir movimientos indeseados. Además instalamos unas ruedas de mayor anchura para establecer una superficie de contacto mayor entre ambas ruedas. Finalmente se soldaron las conexiones y cableado del encoder.

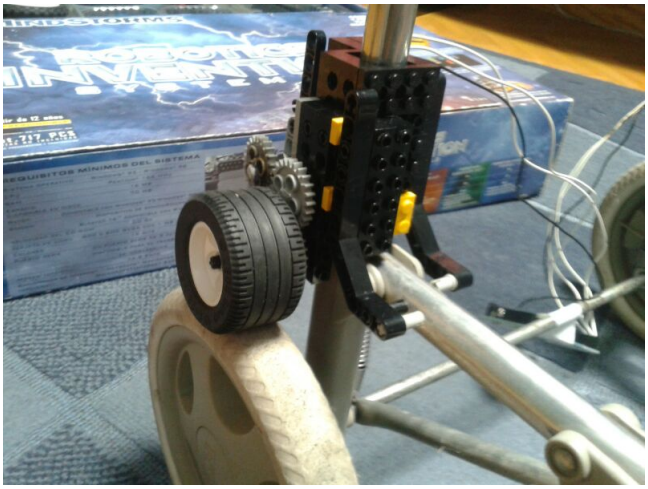

Figura 49: Resultado final del sistema de sujección del encoder

3. Software de reconstrucción y captura de datos

En esta sección se pretende explicar y dejar totalmente claro cada uno de los aspectos de la arquitectura de este proyecto. Se va a realizar por tanto un análisis desde diferentes puntos de vista que permitan reflejar todos los detalles de la misma. El objetivo de cada una de las subsecciones siguientes es:

- **Vista lógica:** Esta sección describe las partes mas significativas de la arquitectura, dividida en una serie de paquetes. Cada paquete se descompondrá en las clases más importantes y se describirán los aspectos más relevantes de dichas clases. Se

describirán las relaciones entre los paquetes y las clases implicadas, asi como sus principales cometidos.

- **Vista de procesos:** Esta sección describe la descomposición del proyecto en los diferentes procesos ligeros en los que se divide el sistema. Para cada proceso se describirán sus funciones dentro del sistema, y los aspectos más importantes de cada uno de ellos. Se explicará también cómo se comunican estos procesos para el correcto funcionamiento del sistema.
- **Vista de despliegue**: Esta sección trata de describir las consideraciones que se han de tener en cuenta, a la hora de ejecutar el sistema, y de los elementos que se necesitan "desplegar" para el correcto funcionamiento del mismo.

En la figura 50 se muestra un diagrama de clases que representa el total de la arquitectura del sistema. Esta figura muestra a modo resumen el conjunto de la arquitectura, por si en algún momento el lector tuviera alguna duda pueda consultarla a modo de referencia. En las siguientes subsecciones se irán presentando cada uno de los elementos de forma más detallada.

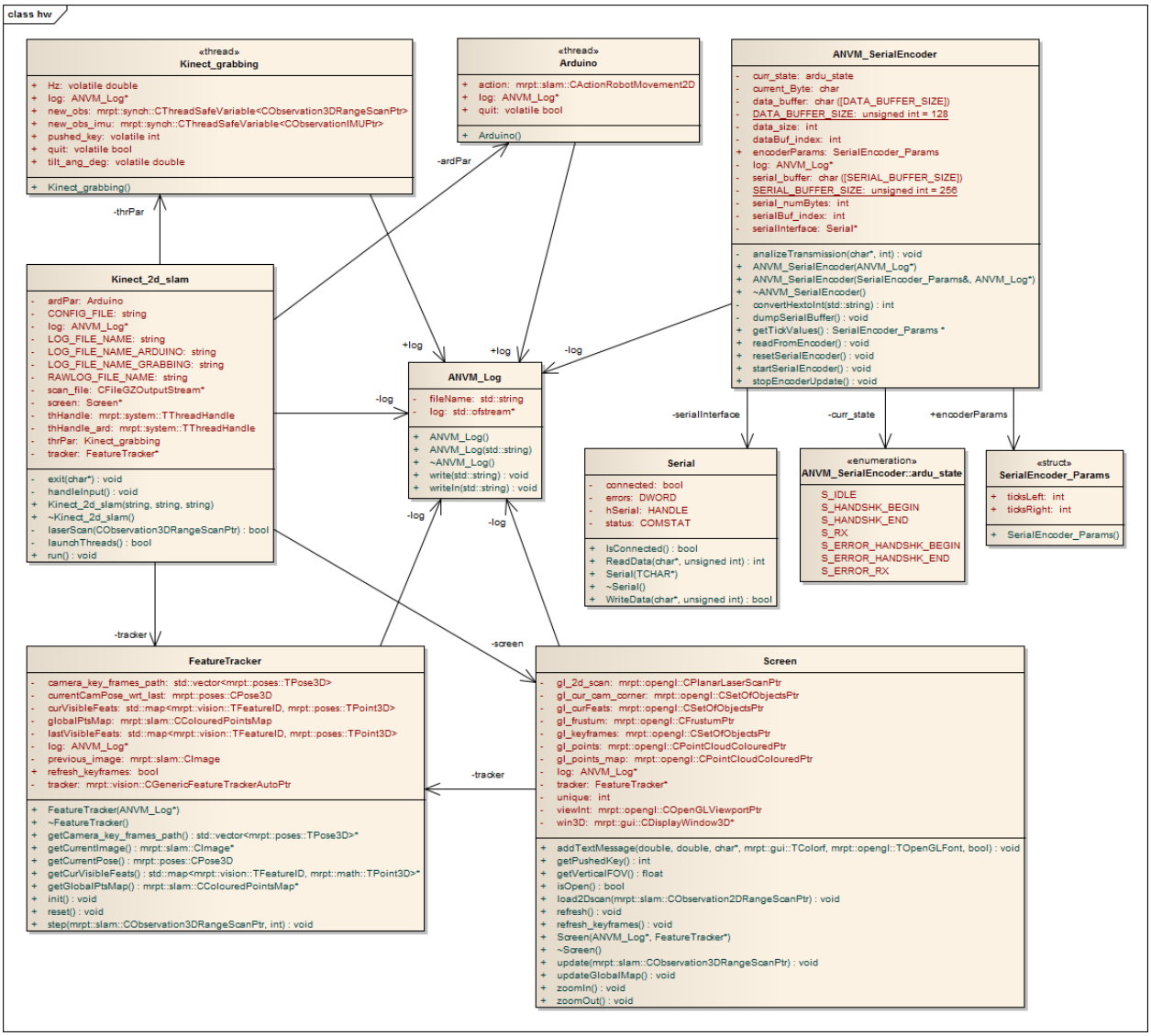

Figura 50: Arquitectura del sistema

3.1. Vista lógica

En esta sección se pretende ofrecer una vista general de la arquitectura del sistema, dividida en cada uno de sus paquetes con una finalidad específica. Para comenzar, se presentan en la figura 51 los paquetes principales de la arquitectura, que corresponden con los objetivos principales de la misma: captura de datos y reconstrucción.

El robot encargado de la captura de datos tiene dos componentes principales: por una parte el dispositivo Kinect que captura información sobre la imagen que se está visualizando y la profundidad de cada uno de los elementos de dicha imagen, y por otra las ruedas que proveen información sobre la posición actual del robot. Por esto, el paquete de captura de datos se divide a su vez en dos sub-paquetes: uno encargado de la recepción de información sobre la imagen proveniente del Kinect, y otro encargado de la recepción de información de la posición u odometría.

El tercer paquete principal que aparece en la figura corresponde con la interfaz gráfica que permite al usuario realizar la reconstrucción de un manera más cómoda. A continuación se describirá el contenido de los diferentes paquetes, así como sus principales clases y funciones.

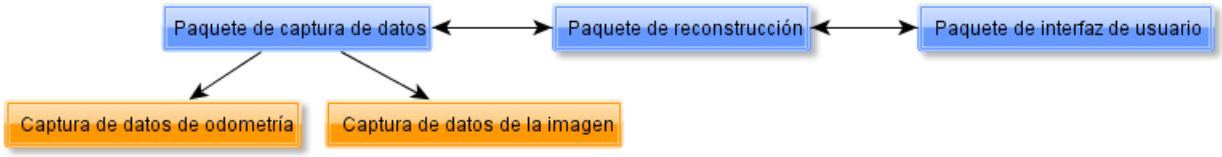

Figura 51: Paquetes principales

Paquete captura de datos de la posición

Este conjunto de clases tiene como misión establecer y mantener la comunicación con el robot que realiza la captura de datos. Esta comunicación se lleva a cabo mediante USB y se sigue un protocolo diseñado e implementado por los desarrolladores del proyecto que se explica en la siguiente subsección en el módulo codificador. Una vez recogidos los datos mediante dicho protocolo, se almacenan y se ponen a disposición del paquete de reconstrucción. En la figura 52 se puede ver un diagrama de las clases que componen este paquete.

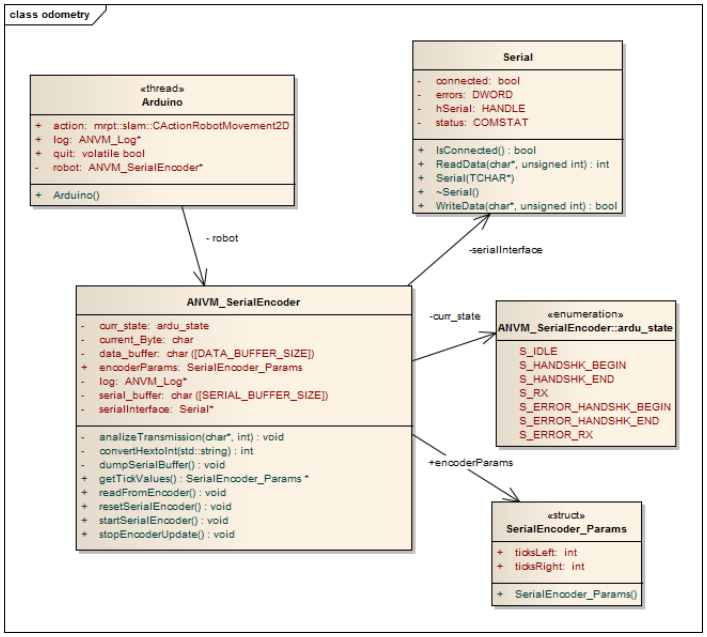

Figura 52: Paquete de captura de datos

Comunicación con el robot de captura de datos

El envío de información entre el microprocesador arduino y el ordenador ejecutando la captura de datos se realiza mediante el paquete de comunicación. Dentro de este paquete encontramos dos módulos (ver figura 53). En primer lugar, el módulo codificador realiza la lectura de codificadores y envío de paquetes de datos desde el microprocesador Arduino [2] hacia el puerto serie del host. Después de esto, el módulo receptor decodifica los paquetes de datos en el ordenador una vez recibidos por el puerto serie. Además tiene la capacidad de resetear el microprocesador Arduino de forma remota.

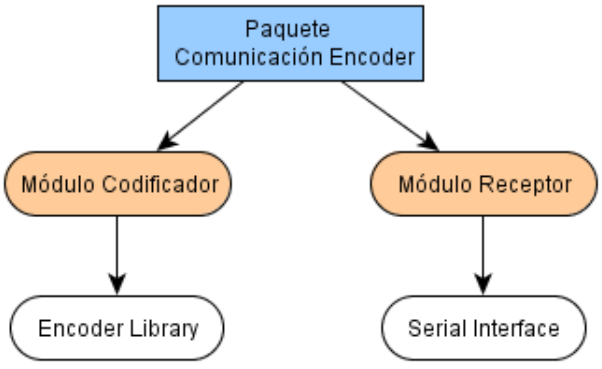

Figura 53: Diagrama de Paquetes: Comunicación con encoders

- **Módulo codificador**

  Se ocupa de contar y almacenar los pulsos de los encoder acoplados a cada rueda. De manera simultánea a esta tarea también empaqueta y envía el número de pulsos almacenados hacia el puerto serie en intervalos de tiempo ajustable. Este módulo se desarrolla en lenguaje Arduino, ya que tiene que ir cargado en dicho microprocesador. Al ser ejecutado en el Arduino hemos de mantener el tamaño del código compilado menor de 32 KB.

  Para el funcionamiento óptimo de los encoders, es esencial que tengan al menos uno de sus dos terminales conectados con una entrada digital habilitada para producir interrupciones. El arduino Uno dispone de las entradas 2 y 3 para este propósito. El mecanismo de pulsos que emplean los encoders para indicar el sentido de la rotación es decodificado mediante la librería *Encoder*, especialmente diseñada para Arduino. Los paquetes de datos están empaquetados como indica la figura 54. Todos los datos numéricos están codificados en hexadecimal y la marca de comienzo de paquete en ascii.

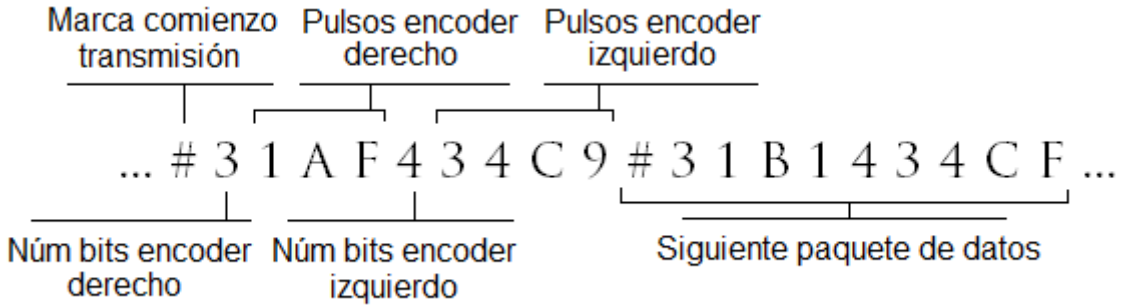

Figura 54: Detalle del paquete de datos enviado por el puerto serie

- **Encoder library**

  Proporciona una interfaz para realizar la implementación del conteo de pulsos por parte de cada encoder. Lo utiliza el módulo codificador para recibir dichos pulsos y poder enviarlos al host. Para un análisis más detallado del funcionamineto de la librería consultar [4].

- **Módulo receptor**

  Se ocupa de consultar periódicamente el puerto serie asignado a la comunicación con el microprocesador arduino, decodificando los datos y almacenando sus resultados. La decodificación consiste básicamente en la agrupación y conversión de bits a hexadecimal de manera eficiente. Ya que el intervalo mínimo de lectura se ve afectado directamente por estas operaciones realizamos una conversión optimizada.

La obtención de paquetes de datos se realiza mediante un puerto serie del ordenador, generalmente el COM3, dicha comunicación emplea la API de Windows para consultar y retirar datos del buffer del puerto. Para facilitar el uso de la API de Windows creamos una interfaz, Serial Interface, que ofrece la misma funcionalidad que dicha API pero encapsulando su complejidad.

El módulo receptor está diseñado para ser ejecutado en su propio hilo de ejecución

- **Serial interface**

  Módulo dedicado a proveer un nivel de abstracción superior a la API de windows para el control de puertos serie. Sus funcionalidades principales consisten en configurar y establecer una conexión a través de un puerto serie, enviar datos por dicho puerto y recibirlos.

- **ANVM_SerialEncoder**

  Se trata de una clase perteneciente al módulo recpetor que se sirve de la interfaz *SerialInterface* para leer del puerto serie y decodificar los datos enviados por el módulo codificador. Sigue el protocolo basado en marcas de comienzo de transmisión que se ha explicado anteriormente en el módulo codificador. Almacena el número de pulsos de cada encoder que va leyendo por el puerto serie.

Almacenamiento de los datos capturados

De esta tarea se encarga principalmente la clase Arduino. Se sirve de la interfaz de comunicación con el robot que proporciona el módulo de comunicación a través de la clase ANVM_SerialEncoder. Este módulo se encuentra en un bucle contínuo que va leyendo cada cierto tiempo (parametrizable) la información que sirve el módulo de comunicación, y la va transformando a información útil para el módulo de reconstrucción.

Esta información que se recibe son el número de *ticks* o pasos que ha dado cada uno de los dos encoders del robot desde que se inició la captura. Al llevar almacenado el número de pasos de la iteración anterior, realiza la resta y obtiene el número de pasos que se han dado en una iteración del bucle. Dicha información sobre el número de pasos que ha dado cada encoder, permite hacer un cálculo estimado de la posición actual del robot con respecto a la anterior. En la sección de Proceso de reconstrucción se explica con más detalle la estimación de la posición. Esta posición se guarda en la variable *action* de tipo *CActionRobotMovement2D* que será recogida e interpretada por el módulo de reconstrucción.

Paquete captura de datos de la imagen

La tarea de este paquete consiste en recibir la información proveniente del dispositivo Kinect. Este paquete está compuesto por un sólo fichero: Kinect_grabbing (ver figura 55). Se realiza una inicialización del Kinect mediante un fichero de calibración, para optimizar la imagen que se va a recibir. Si no se recibe fichero de configuración, se realiza una calibración por defecto. Después de esto, y de la misma manera que el paquete anterior, se ejecuta un bucle contínuo en el que se realiza una lectura en cada

iteración. Esta lectura se lleva a cabo mediante USB, y en este caso se utiliza la librería OpenNI que lleva integrada MRPT para realizar esta comunicación. La información leída en cada iteración del bucle se guarda en la variable "new_obs". Estas observaciones contienen información sobre la imagen que esta capturando en ese instante el Kinect, así como una nube de puntos con la profundidad de los elementos de la imagen respecto del dispositivo. Esta información es la que servirá después al módulo de reconstrucción para construir el mapa.

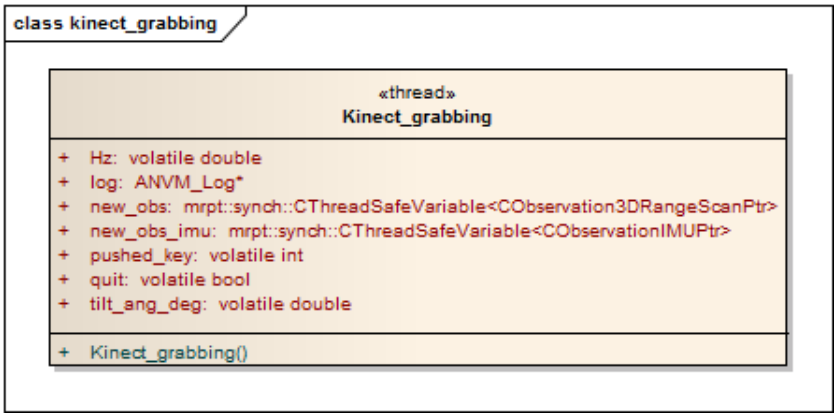

Figura 55: Thread ocupado de leer los datos capturados de la imagen

Como se puede observar este paquete dispone también de un objeto *log*. Permite llevar un seguimiento de los datos que se van leyendo, y para comprobar también que su funcionamiento es el correcto en todo momento. En cuanto al atributo *pushed_key*, sirve como método de comunicación con el paquete de reconstrucción, mediante el cual permite realizar configuraciones dependiendo de la tecla que haya sido pulsada. El atributo *quit* nos permite controlar la ejecución de este thread desde el paquete de reconstrucción.

Paquete de principal de captura de datos

Este conjunto de clases es el encargado de procesar la información capturada por los paquetes anteriores y prepararla para obtener el mapa 2D de la reconstrucción. Es el módulo principal de la arquitectura del sistema, ya que es el encargado de controlar tanto el módulo de captura de datos como el de interfaz de usuario. La clase principal es *Kinect_2d_slam* que coordina todo el sistema de captura. En la figura 56 se pueden observar los principales elementos de este paquete.

Figura 56: Paquete de reconstrucción

Vamos a ir desgranando uno a uno los principales atributos y métodos de la clase Kinect_2d_slam para definir su funcionalidad.

- **thHandle, thrPar**

  Corresponden respectivamente al manejador del thread de ocupado de la captura de imágenes, y el struct de los parámetros de dicho thread. El manejador es un objeto de tipo TThreadHandle que proporciona la librería MRPT y nos permite arrancar el thread, esperar su terminación y comunicarnos con él mediante los parámetros definidios en thrPar. Esta estructura nos permite controlar la ejecución del hilo, mediante el parámetro *quit*, que nos permite indicar al proceso cuando ha de detenerse. También nos permite acceder a la información sobre la imagen que se va actualizando en las variables *new_obs* de dicha estructura.

- **thHandle_ard, ardPar**

  Corresponden respectivamente al manejador del thread de ocupado de la captura de datos de odometría, y el struct de los parámetros de dicho thread. El objeto thHandle_ard es del mismo tipo que el descrito anteriormente, pero en este caso los parámetros son diferentes. La estructura ardPar nos permite consultar la *action* actual, donde está contenida la información sobre la posición del robot. También nos permite controlar su terminación de manera similar al anterior mediante la variable *quit*.

- **tracker**

  Se trata de un objeto FeatureTracker que realiza una estimación de la posición teniendo en cuenta sólo las imágenes provenientes del Kinect. Para realizar esta

tarea se le pasan las observaciones capturadas en cada paso de la reconstrucción, y el objeto tracker (ya de la clase FeatureTracker) se encarga de calcular una estimación de la posición actual que guarda en *currentCamPose_wrt_last*. Este objeto tracker es de tipo *CGenericFeatureTrackerAuto* proporcionado por la librería MRPT. Este objeto se configura mediante una serie de parámetros, y estima internamente la posición teniendo en cuenta la imagen del paso anterior, y la imagen actual. Se puede consultar cómo funciona exactamente este algoritmo, en la sección de Proceso de reconstrucción.

- **laserScan**

  Este método se ocupa de convertir la observación 3D capturada por Kinect a un objeto laserScan2D que simula el haber tomado la imagen con un escáner láser 2D. Se puede consultar cómo se realiza esto exactamente en la sección de Proceso de reconstrucción. Ésta es la información que se almacena, para que construir después el mapa.

- **handleInput**

  Método que permite manejar la entrada proveniente del interfaz de usuario para realizar la acción que se precise en cada momento.

Paquete de reconstrucción

Este paquete se corresponde con el algoritmo Rao-Blackwellised particle filter encargado de generar los mapas a partir de la información capturada. Esta tarea se realiza mediante la implementación de la que dispone la librería MRPT de este algoritmo. Se trata de un archivo ejecutable que recibe como único parámetro un fichero de configuración. El sistema se encarga de realizar la configuración de dicho fichero e invocarlo directamente desde la interfaz como se explica en la sección de Proceso de reconstrucción. Una vez finalizado el algoritmo se genera el mapa construido proveniente del espacio donde hemos realizado la captura de datos.

Paquete interfaz de usuario

Este paquete se encarga de presentar la información al usuario, así como de recoger los datos necesarios para el funcionamiento del sistema. Como se puede observar en la figura 57 existen dos clases claramente diferenciadas: Form1 y Screen. A continuación se explican los detalles de cada una.

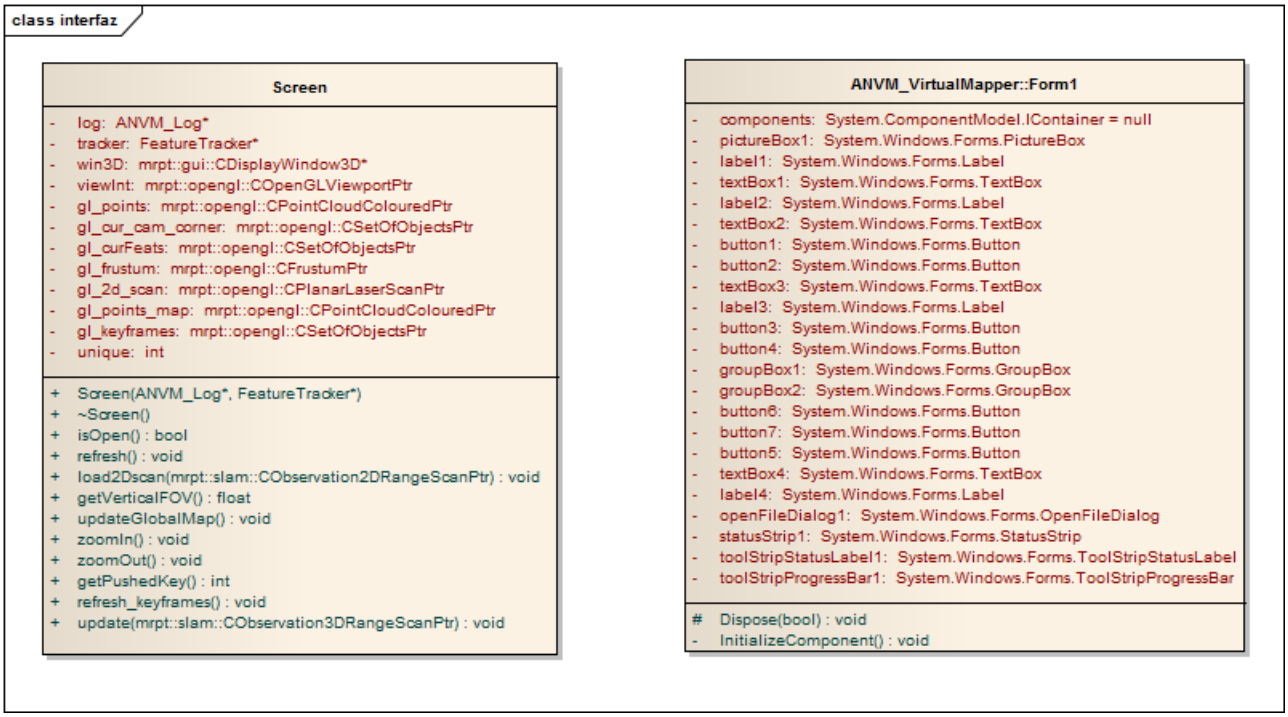

Figura 57: Interfaz de usuario

- **Screen**

  Esta clase está incluida en el mismo ejecutable que el resto de paquetes descritos hasta el momento. Permite visualizar las imágenes que se van capturando con el Kinect a modo de vídeo en tiempo real, así como el láser en dos dimensiones que se está simulando. Se puede ver un ejemplo de captura en la figura 58. Podemos observar el espacio que se está reconstruyendo, y sobre éste, los puntos rojos representan el láser que estamos simulando para conseguir el mapa en 2 dimensiones. En la figura se puede observar que el Kinect detecta con bastante precisión la profundidad de los elementos de las escena (indicada mediante los puntos rojos). También podemos ver que en la parte inferior se nos presentan una serie de datos, así como las letras con las que podemos interactuar con el sistema. Las principales tareas son resetear el sistema, guardar, y salir de la aplicación. A continuación se describen los principales elementos de esta clase:

  - **win3D**: Representa la ventana donde se van a añadir todos los elementos de la escena. Este objeto lo proporciona la librería MRPT y ofrece una forma simplificada de presentar la información de reconstrucción al usuario.

  - **tracker**: Guardamos un puntero a este objeto que nos permite consultar los puntos característicos que se van tomando en cada captura de imagen. Estos puntos se recogen en el atributo *gl_curFeats* que es el objeto que se muestra finalmente en la pantalla.

  - **gl_2d_scan**: Representa el elemento visual del láser que se está simulando. Se permite cargar un nuevo escáner mediante el método *load2Dscan*, utilizado por la clase principal *kinect_2d_slam* para actualizar el láser en cada iteración.

- **getPushedKey**: Permite a la clase principal kinec_2d_slam acceder a las teclas pulsadas por el usuario, y realizar por tanto la acción corresondiente a cada una.

- **update**: Es el método principal que actualiza la vista en la pantalla teniendo en cuenta todos los elementos descritos anteriormente. Recibe como entrada la imagen capturada actualmente por el Kinect.

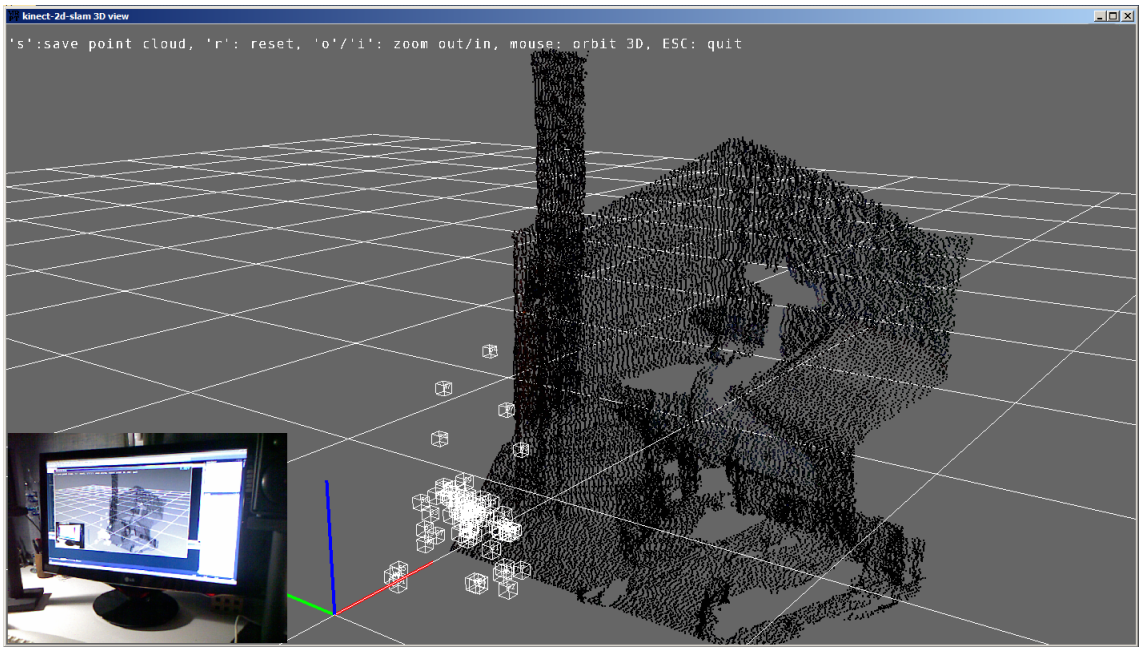

Figura 58: Ventana durante una reconstrucción

- **ANVM_VirtualMapper**

  Se trata de un ejecutable aparte del resto de paquetes comentados anteriormente como se explica en la vista de despliegue (sección 4). Consiste en una interfaz de usuario simple que sirve de primer contacto con el sistema de reconstrucción. En la figura 59 se puede ver una imagen del formulario. En la parte de arriba se muestra una primera sección que nos permite seleccionar aspectos a configurar de la captura de datos, y en la parte de abajo se permite configurar aspectos de la reconstrucción. Esto está explicado con más detalle en la sección de Proceso de reconstrucción.

  En cuanto a los atributos y métodos que observamos en la figura 57 son los propios de una interfaz. Los textbox en los que el usuario introducirá los datos requeridos. Botones para comenzar y terminar una reconstrucción, así como una pequeña barra de carga (toolStripProgressBar1) que se muestra en la esquina inferior derecha de la ventana mientras se está llevando a cabo la reconstrucción.

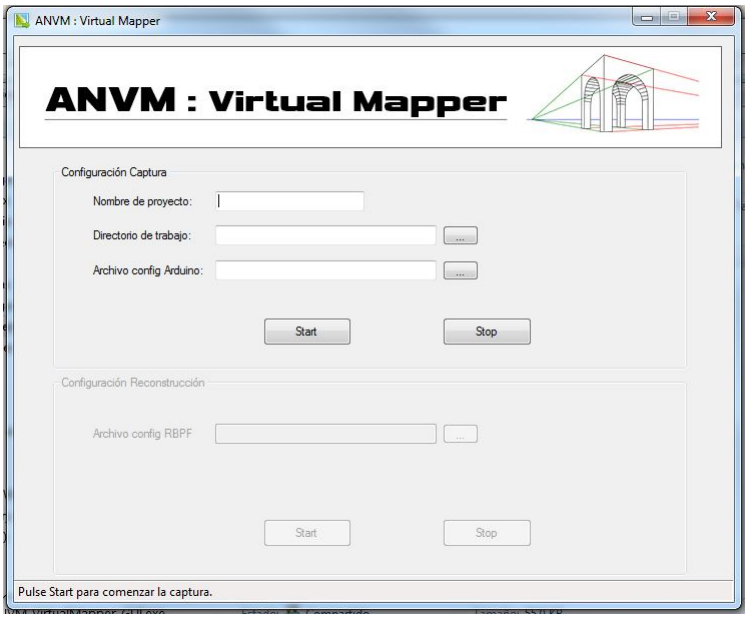

Figura 59: Interfaz inicial Virtual Mapper

La clase Log

Esta clase se utiliza en todo el sitema para llevar un control de la ejecución en todo momento. Como se puede ver en la figura 60 cada objeto de tipo log tiene un fichero asociado donde irá guardando la información necesaria cuando se solicite. Puesto que existen varios hilos de ejecución en este sistema, se crean varios objetos de tipo Log, y por tanto varios ficheros (concretamente uno por cada hilo de ejecución). De esta manera se puede controlar cada proceso del sistema, y en caso de que falle, se puede localizar de manera rápida y sencilla el error que ha llevado a parar la ejecución. Estos ficheros fueron de mucha ayuda durante la implementación ya que eran el único sistema que permitía encontrar errores en el código que se iba desarrollando.

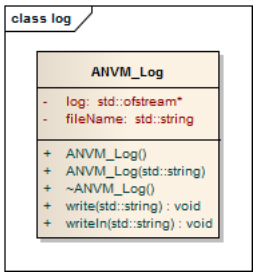

Figura 60: Clase ANVM_Log

3.2. Vista de procesos

Este sistema ejecuta varios hilos al mismo tiempo. Esto es debido a que existen varias tareas que deben realizarse de manera independiente y que necesitan realizar lecturas por USB para comunicarse con el robot de captura de datos. A continuación se van a

comentar los diferentes procesos en los que se divide el sistema, y cómo se comunican entre ellos. En la figura 61 se puede ver un resumen de la comunicación entre dichos procesos.

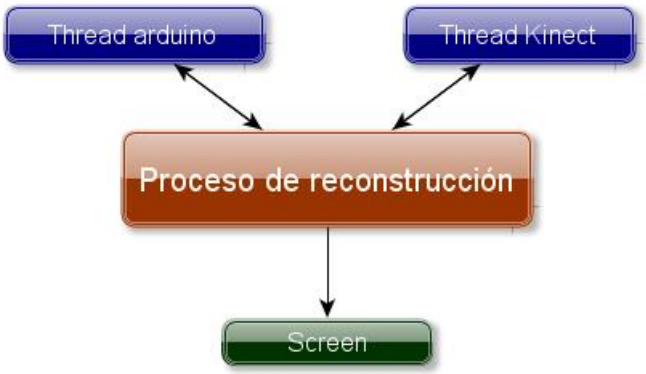

Figura 61: Procesos del sistema

Proceso de reconstrucción

Se trata del proceso principal del sistema que se encarga de arrancar el resto de procesos. Arranca en primer lugar el thread del Kinect, y a continuación el thread Arduino y les asigna un fichero de log diferente a cada uno. Tras arrancar estos thread de captura de datos, arranca la interfaz en la que se va a presentar la información que corresponde con el atributo *screen* descrito anteriormente. Una vez ha iniciado todos los procesos entra en un bucle en el cuál va leyendo de los procesos Arduino y Kinect la información sobre la odometría y la imagen respectivamente, y lo va almacenando en un fichero. En cada vuelta del bucle actualiza también el proceso Screen para visualizar las imágenes que está mostrando el Kinect así como el resto de datos de la reconstrucción.

Thread Arduino

Este proceso se encarga de leer datos sobre la odometría del robot en cada paso de su bucle interno. Se puede configurar para que se lea cada cierto tiempo de el USB, y se va almacenando en la variable “action” la información que se ha leído. La forma de comunicación con el proceso de reconstrucción consiste en tener una estructura de parámetros a la que pueden acceder ambos procesos. Una vez que el proceso de reconstrucción necesite consultar la nueva posición del robot, no tiene más que consultar la variable *action* que este proceso ha dejado actualizada en esta estructura. Cuando se desee la terminación de este hilo basta con cambiar el parámetro *quit* a *false*.

Thread Kinect

Este proceso se encarga de leer datos sobre la imagen del robot en cada paso de su propio bucle interno. La forma de comunicación con el proceso de reconstrucción es similar al thread arduino, mediante una estructura de parámetros. Las imágenes se guardan en

las variables de observación que después consultará el proceso de reconstrucción. Toda la comunicación necesaria entre estos dos procesos se lleva a cabo mediante dicha estructura. Una vez que se recoge la petición de que se desea terminar en el proceso de reconstrucción, se actualiza la variable *quit* de este proceso y finaliza su ejecución.

Screen

Este hilo es controlado por la librería MRPT y permite visualizar los elementos que se van recogiendo en el proceso de reconstrucción en tiempo real. Corresponde con la clase screen mencionada en la anterior sección vista lógica. En cada paso del bucle de reconstrucción se le indica a este hilo qué elementos tiene que mostrar. Cuando se recoge la información de que se desea finalizar la ejecución se elimina el objeto y por tanto el hilo de ejecución.

4. Vista de despliegue

En esta vista se pretenden indicar los elementos necesarios para ejecutar el sistema. Se trata por tanto de dejar claro los elementos que contiene el sistema y expresar su funcionalidad principal, pero en ningún caso explicar detalladamente la tarea de cada elemento. Si se desea conocer los detalles de cada uno de los algoritmos o valores parametrizables de cada configuración se puede consultar la sección de Proceso de reconstrucción donde está todo detallado. Cabe destacar que existen tres ficheros ejecutables que conforman el sistema:

- ANVM_VirtualMapperGUI.exe: Corresponde con el ejecutable de la interfaz de usuario que presenta el sistema. Arranca el sistema y tras una pequeña configuración permite empezar con la captura de datos.
- ANVM_VirtualMapping.exe: Corresponde al código que se ha explicado en la vista lógica, y se invoca al pulsar el botón comenzar de la interfaz anterior. Necesita como argumentos para arrancar las configuraciones realizadas en la anterior interfaz.
- rbpf-slam.exe: Este ejecutable contiene el código necesario para generar el mapa en dos dimensiones con los datos capturados. Es invocado al finalizar la captura, y necesita un fichero de configuración para llevar a cabo la reconstrucción.

Por último es necesario disponer de una serie de .dlls que se distribuyen con la librería MRPT y que dan soporte a las aplicaciones creadas con esta librería. También es necesario disponer del robot construido por el equipo de ANVM-Virtual Mapping capaz de capturar los datos así como los ficheros .ino que cargan el programa de captura de datos en la placa Arduino Uno.

Capítulo 8

Plan de pruebas

En este capítulo de la memoria se profundiza sobre las pruebas y resultados obtenidos en las virtualizaciones. Se han realizado pruebas en diferentes lugares para comprobar las ventajas e inconvenientes del sistema en cada uno de ellos, así como la diferencia en la calidad de los resultados obtenidos dependiendo del lugar donde se realizara la prueba. Las palabras, siglas y acrónimos que necesiten una mayor explicación están recogidas en la sección Glosario.

1. Requisitos de las pruebas

Todas las pruebas que aparecen en esta sección se han realizado con el robot de reconstrucción. Este robot se compone de ciertos elementos mecánicos en conjunción con aparatos electrónicos que se encargan de tomar información del lugar y procesarla para conseguir las reconstrucciones. Estos son los principales componentes del robot de reconstrucción:

- **Dispositivo Kinect de Microsoft:** gracias a las cámaras y sensores infrarojos que posee, este aparato es el encargado de capturar la información 3D de la escena, abarcando un campo de visión horizontal de 57 º y 47º vertical, siendo su rango de profundidad máxima aproximadamente 4 metros.

- **Encoders rotacionales:** se utiliza uno por cada rueda del robot. Producen 15 pulsos por cada giro de 360º que son empleados para la estimación de la posición relativa del robot cuando se están tomando los datos.

- **Placa Arduino Uno:** se encarga de realizar la lectura de los encoders y envío de paquetes de datos desde el microprocesador Arduino hacia el puerto serie del host (el ordenador realizando la reconstrucción).

- **Plataforma móvil:** encargado de transportar el Kinect y la placa Arduino. Consta de dos ruedas sobre las que se apoya el sistema de acoplaje de encoders. Estas ruedas hacen girar el mecanismo sobre el que trabaja cada encoder.

Además, un ordenador portátil conectado al Kinect y al Arduino se encarga de procesar y almacenar los datos para su posterior uso. En este ordenador debe estar cargado el programa de reconstrucción y captura de datos, junto con los archivos de configuración necesarios. El lector se puede consultar en la sección Arquitectura y en la de Proceso de reconstrucción para profundizar detalladamente en cada uno de los componentes que se han descrito y la construcción completa del robot. En la figura 62 podemos ver una imagen del robot de reconstrucción empleado para las pruebas.

Figura 62: Robot de reconstrucción empleado para las pruebas

2. Objetivos de las pruebas

El objetivo principal de todas las pruebas realizadas es la optimización y depuración de resultados para mejorar la precisión del robot. Cada prueba realizada ha servido para detectar errores, hallar patrones satisfactorios que verifiquen la exclusión de nuevos errores en posteriores pruebas y para descubrir comportamientos inesperados. Nos han permitido también diseñar nuevas mejoras que incluir al robot para la generación de mapas de mayor calidad.

3. Pruebas

A continuación se entra más en detalle en cada una de las pruebas concretas que se han realizado. Pueden englobarse dentro de tres grandes hitos que marcaron el desarrollo de las mismas. En primer lugar se utilizó nada más que el Kinect para la virtualización, obteniendo unos resultados muy mejorables. Tras ello, se decidió incorporar información de odometría. Estos datos adicionales se obtienen mediante el cálculo de la posición del vehículo según el movimiento de las ruedas durante el trayecto. La odometría la proporciona parte del conjunto de aparatos que forman el robot capturador, en partiular los encoders y el microcontrolador arduino. Más adelante se decidió incorporar algunas mejoras mecánicas al robot para conseguir una mayor precisión. En los siguientes puntos se profundiza más sobre cada una de estas pruebas.

3.1. Primera versión: Kinect sin uso de odometría

Para las primeras pruebas no se cuenta con el robot como tal, únicamente se utiliza el dispositivo Kinect. La técnica empleada para reconstruir el mapa 2D del lugar, a partir de las capturas del Kinect, fue el algorimo ICP (Iterative Closest Point) que básicamente intenta cuadrar cada captura consecutiva teniendo en cuenta los puntos en común de ambas capturas. Al principio se utilizó esta técnica implementada en la librería MRPT según el consejo de los propios desarrolladores de la librería. Al no disponer de información de la posición en un primer momento, esta era la única alternativa en la librería que permitía realizar reconstrucciones. Esta solución daba resultados aceptables en ubicaciones con muchos rasgos distintivos como esquinas, curvas y obstáculos en general. Sin embargo, al intentar reconstruir zonas lisas y uniformes como, por ejemplo, un pasillo largo o una gran pared, el algoritmo no tiene forma de conocer el desplazamiento del sensor de una captura a la siguiente, ya que al ser la superficie uniforme no hay cambios por lo que no existen rasgos distintivos en la escena y no es capaz de estimar una posición correcta.

Por tanto esta técnica producía unos resultados muy pobres al no disponer de datos de odometría. En la figura 63 se muestran los resultados de estas pruebas. Vemos en la imagen de la izquierda que la reconstrucción comienza bien. El objeto rojo representa el robot, la proyección roja los elementos que se están visualizando en el momento, y los puntos azules los elementos del mapa capturados anteriormente. En esta primera imagen se toma parte de una pared y la esquina de la misma. Sin embargo al avanzar con el robot (imagen de la derecha) el carro detecta muchos elementos en la escena de diferentes profundidades. Al no tener información sobre la posición intenta juntar los puntos que el alogoritmo ICP cree que son similares y se confunde constantemente. Comienza a liar los objetos y a montarlos unos sobre otros sin saber hacia dónde está mirando el robot. A pesar de lo que se comentó anteriormente de que el algoritmo ICP funciona mejor sobre las esquinas (cosa que es cierta) ni siquiera tiene un funcionamiento adecuado cuando se juntan varios objetos de diferentes profundidades en la misma captura de la escena. Por ello esta técnica se descartó rápidamente y se decidió construir un robot capaz de captar información sobre la posición actual.

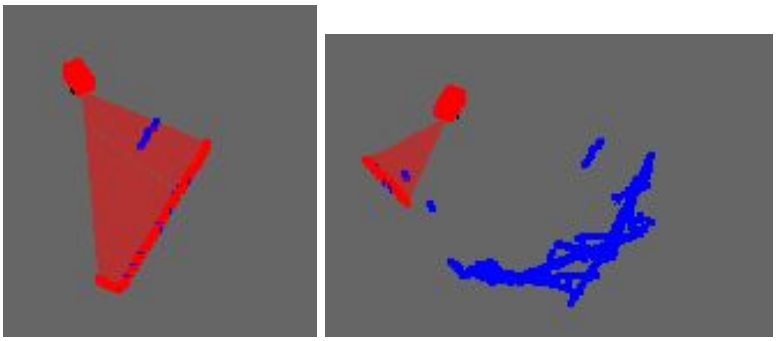

Figura 63: Pruebas ICP slam sin odometría

3.2. Segunda versión: inclusión de odometría

Tras conseguir unos resultados algo decepcionantes, se decide aportar cálculos de odometría que ayuden al procesamiento llevado a cabo por el Kinect. Como se ha comentado anteriormente, la técnica consiste en agregar datos de posicionamiento de forma incremental a las capturas del sensor Kinect para mejorar la reconstrucción global del emplazamiento. Para obtener estos datos adicionales hicimos uso de un vehículo con ruedas equipadas con sensores de rotación para monitorizar su movimiento. Se optó por aplicar esta técnica para mejorar los resultados de la virtualización debido a que proporciona buena precisión, permite tasas de muestreo muy altas y es barata de implantar frente a otras alternativas como acelerómetros y giroscopios.

Sin embargo la principal desventaja de esta solución es que al contener algunos elementos mecánicos, sin un acabado profesional, conlleva una inevitable acumulación de errores a lo largo del tiempo. En concreto, la acumulación de errores en el conteo del número de vueltas que dan los encoders, causa grandes errores en la estimación de la posición, los cuales van aumentando proporcionalmente con la distancia recorrida por el robot.

El cáculo de la odometría está basada en unas simples ecuaciones descritas en la sección de Proceso de reconstrucción, utilizan como entrada los datos que proporcionan los encoders. Debemos recordar que cálculos son válidos bajo la suposición de que el giro de las ruedas sea perfecto. En condiciones reales puede darse el caso de falta de tracción dando lugar a patinazos de las ruedas y otros contratiempos físicos que alteran la correcta estimación de posicionamiento.

Para esta segunda tanda de pruebas se pasó a utilizar el algoritmo Rao-Blackwellised Particle Filter (RBPF) SLAM descrito en la sección Proceso de reconstrucción. Este algoritmo utiliza la información sobre la posición para orientar las capturas de imagen del mapa y unirlas con unos mejores resultados. En la figura 64 se puede observar una captura de un mapa generado con esta técnica. La imagen de la izquierda es el mapa construido por el algoritmo, y la imagen de la derecha es el mapa real de la habitación que se estaba reconstruyendo. El espacio en gris indica espacio no reconstruido, mientras que el espacio en blanco es el espacio interior reconstruido y las líneas negras que delimitan un espacio del otro son las paredes y elementos sólidos de la escena. En la imagen de la izquierda se puede observar que falta la parte superior izquierda de la habitación, dado que no se capturaron datos de esa parte con el robot. Sin embargo se puede apreciar que la forma del mapa construido es muy similar a la del mapa real. Algunas líneas no son del todo rectas como serían las paredes del mapa real, pero en general se consiguió un salto cualitativo bastante alto con respecto a las pruebas anteriores.

3.3. Tercera versión: mejoras físicas en el robot

Tras la inclusión de la odometría en el punto anterior se observó, durante las pruebas, que las ruedas que actúan sobre el sistema de acoplaje del encoder a veces no ejercían su cometido correctamente, debido a que dichas ruedas no eran completamente circulares por el desgaste sufrido en algunas de sus secciones. Esto daba lugar a contar más pulsos

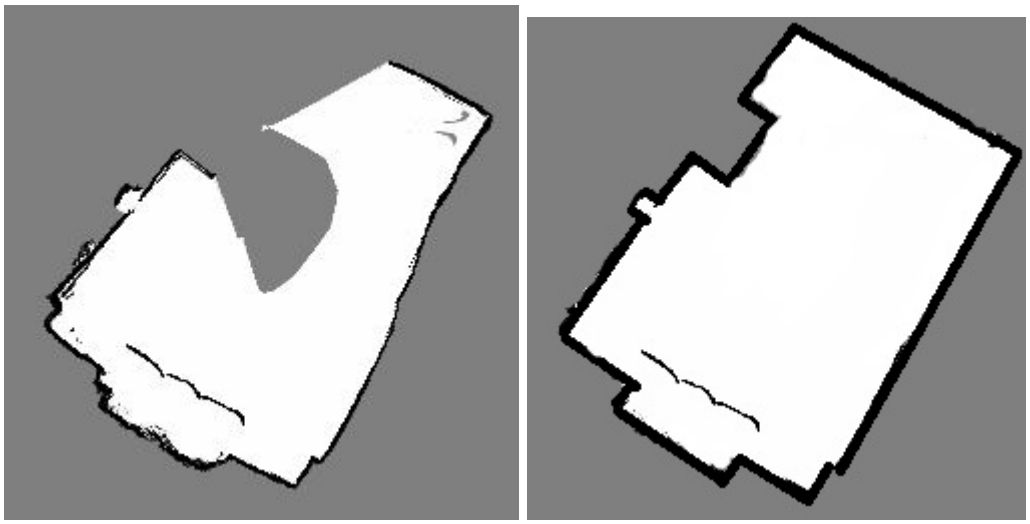

Figura 64: Pruebas RPBF slam con odometría (izquierda). Mapa ideal (derecha).

en un lado que en el otro y al procesar la odometría este desajuste se refleja como un giro por parte del robot cuando en realidad se supone que circulaba en línea recta. Por esta razón se decide realizar aún más mejoras mecánicas al robot.

En primer lugar, se adquieren unos muelles de alta calidad para asegurar que no se deformen durante su labor. Estos muelles se colocan sobre la estructura que sujeta el mecanismo de los encoders como se puede ver en la figura 65.

Figura 65: Muelles y su posterior colocación en el robot

Su función es ejercer una presión constante sobre las ruedas superiores para que tengan un mayor contacto con las ruedas de la plataforma móvil y así se correspondan los giros de ambas y se puedan contar los pulsos correctamente sin saltos inesperados.

Además de la colocación de los muelles, también se sustituyen las anteriores ruedas superiores por unas más anchas y con mayor agarre para mejorar aún más los giros de las ruedas que se mencionaron antes. Esto permite asegurar más superficie de contacto y por tanto las ruedas giran mejor y no se pierden tantos pulsos de los encoders como se perdían anteriormente. En la figura 66 se observan las diferencias entre ambas ruedas.

En la figura 67 se pueden observar los mapas generados con esta técnica tras las mejoras incorporadas al robot, y podemos ver como los resultados obtenidos han mejorado considerablemente. En la figura se observa una sección de un pasillo de la cuarta planta de la Facultad de Informática de la Universidad Complutense. La reconstrucción del

Figura 66: Comparación de las ruedas superiores

mapa comenzaría a la izquierda de la imagen, continuando recto por el pasillo hasta llegar a la esquina final donde se gira a la derecha y se finaliza la reconstrucción. Los cuatro elementos que salen a la izquierda del pasillo corresponden con ventanales y pasillos de despachos por los que el carrito no llegó a pasar. Las dimensiones del espacio reconstruido son mucho mayores a las anteriores pruebas y se obtienen resultados más estables (el mapa es más fiel a la realidad), lo que supone un gran avance.

Figura 67: Mapas generados con el robot mejorado

4. Conclusiones

Se han realizado un gran número de pruebas sobre el sistema de reconstrucción, aquí se recogen tan sólo algunos datos más importantes sobre estas pruebas. Cabe destacar que gracias a las pruebas realizadas se ha podido mejorar el sistema de reconstrucción. En cada prueba se identificaban los errores tanto mecánicos del robot como de implementación que pudiera tener el software de reconstrucción y se han ido perfeccionando ambos.

Han sido clave sin duda también las aportaciones del director de proyecto Juan Carlos Fabero. Tras cada prueba que se realizaba se comentaban los resultados con el profesor, y éste aportaba nuevas ideas y mejoras mecánicas que añadir al robot de reconstrucción. A pesar de que la construcción del robot ha sido llevada a cabo íntegramente por los desarroladores del proyecto, estas indicaciones se tuvieron muy en cuenta y favorecieron la obtención de mejores resultados.

Gracias a todo esto logramos identificar exáctamente los problemas que afectaban al proceso de reconstrucción. Resolviendo estos problemas conseguimos influir directamente en el resultado final de la reconstrucción que es la generación del mapa, perfeccionándolo hasta obtener un nivel de fidelidad adecuado a las necesidades del proyecto.

Capítulo 9

Glosario

En esta sección se define toda la terminología específica relacionada con el desarrollo del proyecto. Se explican las palabras y expresiones que quizá no sean familiares al lector. Algunas de las definiciones que aquí se muestran están basadas en las definiciones de la siguiente dirección web: www.wikipedia.org.

Acg Localizer

Proyecto llevado a cabo por Sattler, Leibe y Kobbelt que permite la localización de una imagen dentro de una reconstrucción de un cierto espacio.

Bin

También llamados contenedores se utilizan en la clasificación de imágenes para describir un histograma en función de sus colores. Cada bin contiene el número de píxeles correspondientes a un cierto rango de colores para un histograma.

BSD License

Licencia de software libre que permite su uso para investigación así como para usos comerciales pero con ciertas restricciones. Éstas dependen del tipo de licencia BSD que se esté considerando.

Bundler

En localización basada en imágenes se corresponde con el proceso SfM.

Clustering, cluster

Hace referencia al proceso de clasificación de un cierto tipo de datos en clases de elementos o clusters.

Dataset

Conjunto de datos de un cierto tipo (imágenes, nubes de puntos...)

Feature Point

Ver Keypoint.

FOV: Field of View

Campo de visión. Aplicado al campo de la visión por computador o informática gráfica se refiere al campo de visión generalmente de una cámara, es decir la región visible (o ángulo de visión) de la que dispone la cámara.

Google

Google Inc. es la empresa norteamericana propietaria de la marca Google, cuyo principal producto es un motor de búsqueda de contenido en internet basado en un algoritmo secreto muy efectivo. A parte del buscador, la empresa ofrece una gran cantidad de servicios web como el correo electrónico Gmail, el mapamundi 3D Google Earth, la mensajería instantánea de Google Talk, el sitio web de vídeos YouTube, el navegador web Google Chrome y su más reciente creación, la red social Google+.

Google Art Project

Se trata de un sitio web promovido por la empresa Google que presenta una recopilación de imágenes en alta resolución de obras expuestas en algunos de los museos más conocidos del mundo. Además cuenta también con un recorrido virtual de las galerías en las que se encuentran dichas obras. La función de exploración de los museos se basa en la misma tecnología usada en Google Street View. Algunos de los museos que cuentan con este servicio son el Tate Britain de Londres, el Museo Metropolitan de Nueva York y la galería Uffizi de Florencia.

Google Maps

Es el nombre de un servicio de Google que ofrece imágenes de mapas desplazables, así como fotos satelitales del mundo. Ofrece también información del tráfico en algunas ciudades y es capaz de calcular recorridos e itinerarios. Es muy parecido a Google Earth sólo que está integrado en la web y no es necesaria su instalación. Cuenta con aplicación propia para Android y iOS a parte de la propia web. El transporte público de algunas ciudades importantes y muchos comercios aparecen al navegar con Google Maps, siendo esta una buena manera de promocionar su negocio y para Google una fuente de enriquecimiento de contenido.

Google Play Store

Google Play (antes conocido como Android Market) es una tienda de software en línea desarrollada por Google para los dispositivos Android. Permite a los usuarios buscar y descargar aplicaciones, libros, revistas, series, películas, música y otros servicios a sus dispositivos con Android. Los usuarios pueden ver las valoraciones de otros usuarios, así como algunas imágenes o vídeos y una descripción del contenido antes de su compra o descarga. Actualmente se han sobrepasado ya las 700000 aplicaciones disponibles en la tienda.

Google Ride Finder

Se trata de una herramienta basada en Google Maps que usando la localización permanente de taxis y limusinas mediante GPS, ofrece al usuario los vehículos disponibles cerca de su ubicación. Fue desarrollada sólo para San Francisco y Chicago en Estados Unidos. A partir de 2009 su soporte fue interrumpido.

Google Sites

Es una herramienta creada por Google estructurada como una wiki o creador de páginas web. El objetivo de Google Sites es conseguir que cualquiera sea capaz de crear un sitio web dónde muchas personas puedan colaborar y compartir archivos e información.

Google Street View

Es una extensión de Google Maps y Google Earth que proporciona imágenes panorámicas (360 grados de movimiento horizontal y 290 grados de movimiento vertical), permitiendo a los usuarios ver partes de las ciudades seleccionadas y sus áreas metropolitanas. Ahora mismo está disponible en 25 países europeos, 3 americanos, 11 asiáticos, 2 africanos y en la Antártida.

Las virtualizaciones se llevan a cabo con láseres de última generación y cámaras especiales incorporadas en un vehículo que recorre la ciudad para su reconstrucción. Toda las fotografías son siempre editadas antes de la publicación final, eliminando caras y matrículas debido a las políticas de privacidad internas de cada país.

Actualmente también cuenta con versiones para dispositivos portátiles y una API gratuita que permite insertar versiones de HTML y ActionScript en una web personal o de empresa, con el resultado de poder visualizar imágenes de Street View con un tamaño y contexto elegidos por el usuario.

Google Traffic

Es un complemento de Google Maps que aporta información al usuario sobre el tráfico de las carreteras. Está disponible para Android y iOS. La información sobre el tráfico se

consigue tomando datos anónimos de la señal GPS de los teléfonos y creando estimaciones.

Histograma

En localización basada en imágenes, se corresponde con una cuadrícula de la imagen formada por un cierto conjunto de píxeles y que se utiliza para la posterior clasificación de esa sección de la imagen.

ICP

Iterative Closest Point algorithm es un algorimo iterativo que consiste en ir analizando pares de puntos de dos conjuntos (nubes de puntos generalmente) para ir reduciendo al mínimo la diferencia entre ambas nubes de puntos. Se utiliza para la reconstrucción tanto 2D como 3D.

Inlier

Elemento de un conjunto (o muestra en RANSAC) que se ajusta correctamente a un modelo.

iOS

iOS es un sistema operativo para móviles desarrollado por la empresa americana Apple Inc. Inicialmente fue desarrollado para el smartphone de la compañia, el iPhone, y más adelante fue implantando en otros dispositivos como iPod Touch, iPad y Apple TV. Apple no permite la instalación de iOS en hardware de terceros.

Keypoint

Punto característico de una imagen (también llamado features points). Es un punto que dispone de alguna característica que le da significado a un objeto de la imagen. El contorno de un objeto por ejemplo estaría representado por un conjunto de keypoints.

Kinect™ sensor de Microsoft™

Es un dispositivo desarrallado por la compañía Microsot™ que permite la captación de nubes de puntos mediante la cámara y el sensor de infrarrojos colocados en su parte delantera.

Kinfu: Kinect Fusion

Proyecto de la librería PCL que permite la reconstrucción tridimensional mediante modelos de un cierto espacio.

LIDAR

Light Detection And Ranging: tecnología de teledetección óptica que puede medir la distancia a objetivos por medio de iluminar el blanco con luz láser y analizando la luz dispersada. LIDAR tecnología tiene aplicaciones en geomática, arqueología, geografía, geología, geomorfología, sismología, etc. Mediciones LIDAR son medidas estadísticas creadas a partir de la nube de puntos 3D adquiridas mediante LIDAR.

Match

Cualquier uso en este documento de la palabra match o sus derivadas (matching, matches...) debe entenderse con el significado que tiene en inglés de encaje o emparejamiento, ya sea entre imágenes, puntos o cualquier otra cuestión.

NFC, NFC tag

Near field communication: es un conjunto de estándares para smartphones para establecer la comunicación por radio entre sí mediante contacto físico o colocándolos en estrecha proximidad, normalmente no más de unos pocos centímetros. Aplicaciones actuales y previstas incluyen transacciones sin contacto, intercambio de datos, etc. La comunicación también es posible entre un dispositivo NFC y un chip NFC sin alimentación eléctrica llamado "tag".

Nube de puntos

Conjunto de puntos en tres dimensiones, es decir, cada punto de la nube dispone de tres coordenadas: x, y, z.

Odometría

Estudio de la estimación de la posición de vehículos con ruedas durante la navegación. Para realizar esta estimación se usa información sobre la rotación de las ruedas para estimar cambios en la posición a lo largo del tiempo.

OPENNI

Librería de software libre sobre la que se apoya PCL y otras muchas para recibir información de dispositivos capaces de captar nubes de puntos.

Outlier

Elemento de un conjunto (o muestra en RANSAC) que no se ajusta a un modelo.

PCL: Point Cloud Library

Librería que permite el manejo y captación de nubes puntos.

Photo Tourism

Photo Tourism es un sistema para la navegación de grandes colecciones de fotos en 3D. El sistema toma como entrada colecciones de fotos de usuarios subidas a sitios públicos de internet y automáticamente procesa cada foto y cada punto de vista y reconstruye la escena en tres dimensiones. El explorador de fotos permite al usuario moverse interactivamente por el entorno 3D realizando transiciones entre las fotografías.

QR: Quick Response code

Un código QR (del inglés, código de respuesta rápida), es un módulo útil para almacenar información en una matriz de puntos creada por la empresa japonesa Denso Wave en 1994. Se caracteriza por los recuadros que aparecen en 3 de las esquinas que se utilizan para detectar la posición del lector.

Aunque inicialmente sólo se usaban en la industria, la inclusión de lectores en dispositivos móviles como tablets y smartphones ha permitido que su uso se oriente hacia el consumidor. Se busca facilitar el día a día a los usuarios y que estos no tengan que introducir información manualmente en el teléfono y puedan hacerlo a través del lector de QR. Es muy común encontrarlos en revistas y cartelería conteniendo las URLs de los anunciantes. Cada vez más se está utilizando también en tarjetas de visita, almacenando todos los datos del interesado. La capacidad de almacenamiento de un código QR puede ser de hasta 4296 caracteres alfanuméricos.

Cabe destacar por último que a diferencia de otros formatos de códigos como Bidi, el código QR es abierto y sus derechos de patente no son ejercidos.

RANSAC

Abreviatura de “RANdom SAmple Consensus” es un algoritmo iterativo que trata de ajustar los parámetros de un modelo matemático a patir de unas muestras (samples) que

contienen outliers. Trata de ajustar esas muestras al modelo y es aplicado en multitud de campos, concretamente en este documento se habla de él en la sección del proyecto Acg localizer, donde se utiliza para estimar la posición de la imagen. Este algoritmo itera hasta que se encuentra un cierto número de inliers o se alcanzan un número máximo de iteraciones.

ROS: Robot Operating System

Conjunto de librerías desarrolladas para Linux por multitud de investigadores y científicos del campo de la robótica, que aporta gran funcionalidad para la creación de aplicaciones en este campo.

Sample

Término inglés que se traduce como muestra, y representa un conjunto de elementos que se tienen en cuenta en cada una de las iteraciones del algoritmo RANSAC.

SLAM: Simultaneous localization and mapping

Técnica que consiste en la reconstrucción o virtualización de un espacio a medida que se va avanzando por el mismo. Se presupone que no se tiene ninguna información previa sobre el espacio, antes de realizar la reconstrucción.

SfM: Structure from Motion

Proceso de reconstrucción de la estructura tridimensional de un objeto a través del análisis de información en 2D (imágenes).

VW: Visual Word

En el proyecto Acg localizer hace referencia a un sistema de clustering que permite que permite búsquedas más eficientes.

XML

XML, siglas en inglés de eXtensible Markup Language (’lenguaje de marcas extensible’), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C). Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información. (Bases de datos Silberschatz).

Capítulo 10

Bibliografía y referencias

Este capítulo de la memoria recoge las referencias y fuentes de mayor relevancia durante el desarrollo del proyecto.

Arquitectura

- Componentes de la plataforma de reconstrucción.

  [1] http://www.alps.com/products/WebObjects/catalog.woa/E/HTML/Encoder/Incremental/E

  [2] http://arduino.cc/en/Main/arduinoBoardUno

- Software diseño Lego.

  [3] http://www.leocad.org/

- Librerías Arduino.

  [4] http://www.pjrc.com/teensy/td_libs_Encoder.html

Estado del arte y visión

- Alternativas y competidores.

  [5] http://viametris.fr/

  [6] http://www.matterport.com/

- Aplicaciones y servicios relacionados de Google.

  [7] https://play.google.com/store/apps/details?id=com.google.android.apps.maps

  [8] http://www.google.com/culturalinstitute/project/art-project

- Proyecto hermano ANVM - Mobile App.

  [9] ANVM - Mobile App, Víctor Ortíz García, Miguel Gutiérrez García-Cuevas, Proyecto fin de carrera 2013, FDI, UCM.

Investigación

- Información sobre el dispositivo Kinect.

  [10] http://www.xbox.com/en-US/kinect

- Descriptores investigados.

[11] SURF: Speeded Up Robust Features Herbert Bay1 , Tinne Tuytelaars2 , and Luc Van Gool12 1 ETH Zurich {bay, vangool}@vision.ee.ethz.ch 2 Katholieke Universiteit Leuven {Tinne.Tuytelaars, Luc.Vangool}@esat.kuleuven.be

[12] S.M. Smith and J.M. Brady (May 1997). "SUSAN – a new approach to low level image processing". International Journal of Computer Vision 23 (1): 45–78. doi:10.1023/A:1007963824710.

- Diferentes implementaciones de Structure-From-Motion.

[13] http://www.cs.cornell.edu/~snavely/bundler/

[14] http://homes.cs.washington.edu/~ccwu/vsfm/

- Librerías y Proyectos investigados.

[15] http://pointclouds.org/

[16] http://www.rwth-graphics.de/software/image-localization

[17] http://www.ros.org/

[18] http://www.mrpt.org/